UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERIA DE SISTEMAS

Tesis previa a la obtención del título de: INGENIERO DE SISTEMAS

TEMA:
SISTEMA DE CONTROL DE INVENTARIO EN ENTORNO WEB Y DISPOSITIVOS MÓVILES CON SISTEMA OPERATIVO ANDROID PARA LA EMPRESA RODAMIENTOS BOWER

AUTORES:
DANNY GABRIEL QUISHPE PALOMEQUE
PABLO JAVIER NÚÑEZ MOYA

DIRECTOR:
GUSTAVO ERNESTO NAVAS RUILOVA

Quito, abril del 2014

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además declaramos que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.
Quito, Marzo 2014

Danny Gabriel Quishpe Palomeque	Pablo Javier Núñez Moya
CI: 1721857520	CI: 1803313657

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este proyecto tiene como objetivo principal mejorar la forma de cómo se maneja en inventario dentro de la empresa Rodamientos Bower, mediante el uso de lenguajes de programación libres y para dispositivos móviles así como el diseño y desarrollo de una página web se busca brindar las herramientas necesarias para que la empresa pueda controlar todos los ítems que se encuentran en su bodega de forma rápida y manteniendo el mayor control posible.

No solo es un proyecto de desarrollo de habilidades tecnológicas sino sirve para un desarrollo de habilidades de negociación y comunicación entre el desarrollador y el cliente (usuario final del sistema), así como también para mejorar la forma como se emplean metodologías de desarrollo y arquitecturas de software.

Todo lo mencionado anteriormente hace de este proyecto una gran experiencia, muy enriquecedora tanto para las personas encargadas de desarrollarlo como cada uno de los implicados de la empresa.

ABSTRACT

This project's main objective is to help improve the way inventory is handled within the company Rodamientos Bower. By using free programming languages and mobile devices as well as the design and development of a web page aims to provide the tools necessary for the company to control all of the items currently in your wine quickly and keeping the best possible control.

It is not just a development project of technological skills but also helps a lot to develop negotiation and communication skills of the developer of the client (end-user system), as well as improve the way development methodologies and software architectures are used.

The entire make of this project is a great experience, very rewarding the develop people and the people who work in the company.

INTRODUCCIÓN

Cuando Jorge Ramos inicio su empresa manejaba un número controlable de artículos en su bodega, con el paso del tiempo este número fue incrementándose de tal forma que hoy no es controlable de forma manual. En ese momento nace el problema que este proyecto intenta solucionar.

El número actual de productos que se encuentra en la matriz de la empresa es aproximadamente de 60 mil ítems diferentes. Este número tiende a incrementarse anualmente, si sumariamos los ítems que tiene cada una de las sucursales podríamos decir que tiene un aproximado de 120 mil ítems. El control del inventario de la bodega se realiza de forma manual y aunque se hace un gran esfuerzo por tener todo bajo control es casi imposible tener un verdadero control de la bodega.

Uno de los problemas principales que se presentan al momento de querer implementar un software de estas características en esta empresa es migrar todos los archivos que tiene la empresa al sistema, este proceso puede representar una gran pérdida de tiempo y dinero a la empresa, aunque por otro lado el no tomar en cuenta el crecimiento de las bodegas y de la empresa en general puede resultar a la larga una perdida mucho mayor.

La mayoría de operaciones dentro de la empresa se realizan de forma manual, por lo que no se encuentra implementado ningún sistema de control de inventario, tampoco se tiene almacenado datos en alguna base de datos por lo que se deberá ingresar los datos de los productos de forma manual.

El sistema que se desarrollará en este proyecto será la base para los otros sistemas como por ejemplo el sistema contable, aunque se limitará en lo posible su alcance para que realice su labor principal controlar el inventario de la bodega.

Objetivo General:

- Desarrollar e implementar un sistema web y un sistema android para el control del inventario general y por sucursal de la empresa Rodamientos Bower.

Objetivos Específicos:

1. Recolectar los requerimientos de cada uno de los implicados del proyecto enfocándose principalmente hacia los vendedores y personal administrativo de la empresa.

2. Investigar información sobre la lectura de código de barras mediante dispositivos móviles, así como también desarrollar un sistema que sea capaz de leer códigos de barras desde dispositivos con sistema operativo Andriod.

3. Diseñar y generar la base de datos necesaria para el funcionamiento del sistema a desarrollarse, tomando en cuenta todos los posibles cambios que se puedan realizar a lo largo del desarrollo del sistema.

4. Migrar datos de los productos, sucursales, usuarios, etc desde los archivos impresos de la empresa a la base de datos y realizar pruebas funcionales del sistema con estos datos.

5. Diseñar una aplicación web de control de inventario que se adapte a las necesidades de la empresa, así como también que cumpla con las especificaciones de una arquitectura orientada a servicios.

6. Diseñar y desarrollar una aplicación móvil que sea capaz de comunicarse con el sistema web utilizando servicios web para realizar la comunicación.

7. Realizar diferentes tipos de pruebas en el sistema y verificar que el mismo se encuentra funcionando por encima de los niveles de aceptación de la empresa Rodamientos Bower.

Justificación del proyecto

Rodamientos Bower es una empresa con un gran número de ítems en bodega, por lo que realizar un control de inventario de forma manual es muy complejo, por eso la necesidad de desarrollar este sistema mediante el cual ayudará a mantener el control de todos los ítems que se encuentran en la bodega, evitando posibles pérdidas para la empresa sea por extracción no autorizada de ítems o por no tener suficiente stock en el momento de la venta.

Este proyecto también aportara beneficios a la persona encargada del desarrollo del mismo, no solo por la enriquecedora experiencia de manejar un proyecto real, sino por el conocimiento que se adquirirá al desarrollar dicho proyecto, al con nuevas tecnologías.

El uso de dispositivos móviles incrementa día a día, es así como en la actualidad la mayoría del tráfico de internet a nivel mundial lo ocasionan tablets y los teléfonos celulares inteligentes haciendo que la información sea accesible desde cualquier lugar, dejando a un lado la necesidad de computadores de escritorio.

Esto también conlleva a que las aplicaciones de una empresa poco a poco vayan migrando de un sistema de escritorio a uno móvil, ya que la mayoría de los empresarios busca consultar información necesaria de su empresa de forma inmediata y en cualquier lugar.

Alcance del proyecto

El software de este proyecto será diseñado para realizar un control de inventario, por lo cual el sistema controlara principalmente el ingreso y salida de productos.

El ingreso y la salida de productos de la bodega se podrán realizar únicamente a través del sistema web, manteniendo un control de la sucursal en la que se realiza algún cambio. Los roles de usuario que tendrán acceso a las opciones serán el administrador y el usuario de bodega, mientras que para la salida de productos podrá realizar el rol vendedor, bodega y el administrador.

La salida de los productos se podrá realizar de dos formas, por medio de la pantalla de venta de productos y por la pantalla de administración de productos en bodega. La pantalla de venta de productos fue solicitado por la empresa, pero esta pantalla no entra dentro del alcance del proyecto, se construirá una pantalla básica para vender productos principalmente para realizar el control de la salida de los productos y no para ser tomado como parte de un sistema de ventas. La pantalla de venta de productos no emitirá ningún tipo de factura o nota de venta y no tendrá mucha de las funciones que tendría una pantalla de venta de productos.

El sistema de lectura de código de barras se aplicará solamente en el sistema móvil, esto es principalmente por pedido de la empresa Rodamientos Bower, porque no quiere implementar un sistema de lectura de código de barras. El código de barras de cada producto podrá ser ingresado por la persona encargada del ingreso de productos, pero será decisión de la empresa si utiliza los códigos de barras generados por la fábrica de cada uno de los productos comercializados.

El sistema móvil se puede utilizar solamente en los tablets con sistema operativo android que tengan como mínimo la versión 4.0, el sistema podrá ser utilizado por cualquier dispositivo móvil pero las pruebas del sistema se realizaran en el modelo google nexus 7.

CAPÍTULO 1

INFORMACIÓN GENERAL DEL PROYECTO

1.1 Información de la empresa rodamientos Bower

Rodamientos Bower es una empresa dedicada a la compra-venta de rodamientos y retenedores. Esta empresa tuvo sus inicios en Ambato, fundada por Jorge Ramos actual gerente de la empresa, esta inicio con una sola sucursal la cual es la oficina principal.

La empresa en un inicio solo distribuía rodamientos, luego se agregó a su inventario otros tipos de producto como por ejemplo retenedores. La bodega comenzó con un promedio de 500 a mil productos, llegando a un máximo de 2000 productos, hoy en día esta cifra aumentó al punto de tener más de 15000 productos diferentes. Esto representaría que en bodega se encuentran almacenados entre 50000 y 60000 items y tomando en cuenta solo al edificio principal. La empresa es uno de los principales distribuidores de rodamientos en Ambato, con 3 sucursales en diferentes zonas estratégicas de la ciudad. Con el tiempo no solamente se limitaron a distribuir sus productos dentro de Ambato, sino que agrandaron sus horizontes a tal punto que tienen 2 sucursales en el valle de Sangolqui y una sucursal en la ciudad de Puyo.

Aunque todas las sucursales pertenecen a la misma empresa, cada una de ellas tiene diferente propietario y gerente, por lo cual se puede decir que se manejan como empresas individuales. Sin embargo, cada sucursal cuenta con el apoyo de las demás y comparten información acerca de su inventario.

1.1.1 Diagnóstico interno de la empresa

1.1.1.1 Limitación geográfica de la empresa

La empresa consta de 4 sucursales y la matriz. El proyecto será implementado en la matriz por lo cual nos enfocaremos directamente en esta localidad.

El “edificio Bower” así nombrado por su propietario se encuentra en la avenida 12 de Noviembre y Unidad Nacional en la cuidad de Ambato.

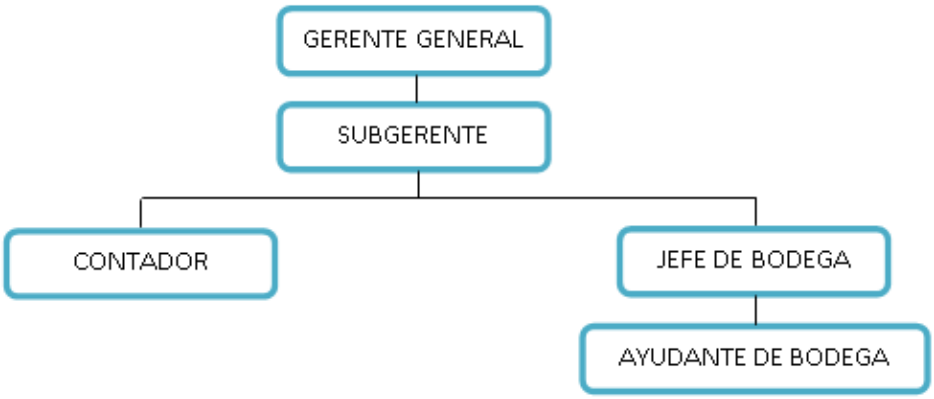
Figura 1 Croquis Edificio Bower (Ambato)

Elaborado por: Pablo Núñez, 2013

1.1.1.2 Organigrama Estructural

Como fue mencionado antes cada sucursal de la empresa tiene su propio dueño y gerente por lo que el organigrama se realizara solamente de la matriz. El organigrama de la empresa se puede visualizar en la figura 2.

Figura 2 Organigrama de Rodamientos Bower



Elaborado por: Pablo Núñez, 2013

1.1.1.3 *Direccionamiento Estratégico*

Visión de la Empresa

"Tener un stock de repuestos que satisfaga las necesidades de la línea de cualquier vehículo o maquinaria existentes en el país"

Misión de la Empresa

"Satisfacer de la mejor manera las necesidades o requerimientos de nuestros clientes"

1.1.1.4 Objetivos de la empresa

Objetivo General

Satisfacer las necesidades de los clientes y de todas las marcas que requieran los mismos a fin de tener un stock completo sobre la línea de rodamientos.

Objetivos Estratégicos

- Incrementar programas de contabilidad, control de estradas y salidas de mercaderías.
- Incrementar dos nuevas sucursales con la capacidad de monitorearles desde cualquier lugar del país.

1.1.1.5 Análisis FODA

Fortalezas

- La empresa es bien conocida dentro de la ciudad de Ambato, una gran ventaja en cuanto a la apertura de sucursales dentro de esta ciudad.
- La empresa tiene un gran número de ítems en su bodega, lo que permitiría levantar una nueva sucursal de forma rápida.
- La empresa tiene un gran tiempo en el mercado, ventaja en cuanto a conocimiento de los mejores distribuidores de los rodamientos así como también de todos los productos que esta empresa comercializa.

Oportunidades

- Los rodamientos son muy necesarios para todo tipo de automotor, principalmente para los de trabajo pesado, lo cual permite el rápido crecimiento de la empresa en rurales.
- El comercio mediante el uso de internet tiene cada día más aceptación por los ecuatorianos, lo cual crea nuevas oportunidades para empresas encargadas del comercio.

Debilidades

- Aunque la empresa es conocida en Ambato es muy poco conocida en otros lugares como por ejemplo Quito, lo cual representa una gran desventaja ante empresas que ya tienen un tiempo funcionando en ese lugar.

Amenazas

- Las páginas dedicadas al comercio de cosas como por ejemplo MercadoLibre.com ocupan un lugar importante dentro del comercio de artículos en general, las personas poco a poco preferirán buscar sus productos en internet ya que esta es una forma más rápida y eficaz de encontrarlos lo cual representa una amenaza para empresas que no adopten esta modalidad.
- Apertura de nuevos almacenes que se dedican a la comercialización de rodamientos o retenedores.

1.2 Importancia de las aplicaciones móviles en el mundo actual

En los últimos años existe un creciente reconocimiento del destacado papel que juegan las comunicaciones móviles en todos los ámbitos de nuestra sociedad. Las comunicaciones móviles nos acompañan en nuestros quehaceres diarios y también son uno de los principales medios para que la economía sea más dinámica y eficiente. Es más, a diferencia del desconocimiento que impulsó algunas de las exageradas especulaciones típicas del pasado reciente, sabemos ahora que, precisamente por su profunda relación con las mismas actividades básicas de la economía y la sociedad, también el contexto cultural, social, político, económico en el que se han desarrollado las comunicaciones móviles en un país concreto ha tenido una enorme influencia en su devenir. Hay, por tanto, un camino de ida y vuelta entre comunicaciones móviles y sociedad, en el que mutuamente se influencian y modifican, y que depende de las características concretas de esa sociedad.

Sin embargo, aunque existen numerosas y notables contribuciones sobre diferentes aspectos parciales de la ‘apropiación social' de las comunicaciones móviles, no existe un examen general de su impacto social, incluyendo una

perspectiva histórica y una evaluación de las implicaciones de los próximos desarrollos. (fundaciontelefonica, 2011)

1.3 El internet y las redes móviles

La primera descripción registrada de las interacciones sociales que se podían habilitar a través de la red fue una serie de memorandos escritos por J.C.R. Licklider, del MIT, en agosto de 1962, en los que describe su concepto de "Red galáctica". Imaginó un conjunto de ordenadores interconectados globalmente, a través de los que todo el mundo podría acceder rápidamente a datos y programas desde cualquier sitio. En espíritu, el concepto era muy similar a la Internet de hoy en día. Licklider era el director del programa de investigación informática de DARPA,[4] que comenzó en octubre de 1962. Mientras estaba en DARPA convenció a sus sucesores en dicha agencia (Ivan Sutherland, Bob Taylor y Lawrence G. Roberts, investigador del MIT), de la importancia de su concepto de red.

Leonard Kleinrock, del MIT, publicó el primer documento sobre la teoría de conmutación de paquetes en julio de 1961 y el primer libro sobre el tema en 1964 Kleinrock convenció a Roberts de la factibilidad teorética de comunicarse usando paquetes en vez de circuitos, lo que fue un gran paso en el viaje hacia las redes informáticas. El otro paso clave fue conseguir que los ordenadores hablasen entre sí. Para explorar esta idea, en 1965, trabajando con Thomas Merrill, Roberts conectó el ordenador TX-2, en Massachusetts, con el Q-32, en California, mediante una línea telefónica conmutada de baja velocidad, creando la primera (aunque pequeña) red de área amplia del mundo. El resultado de este experimento fue la constatación de que los ordenadores con tiempo compartido podían trabajar bien juntos, ejecutando programas y recuperando datos según fuese necesario en el equipo remoto, pero que el sistema telefónico de conmutación de circuitos era totalmente inadecuado para esa tarea. Se confirmó la convicción de Kleinrock de la necesidad de la conmutación de paquetes.

A finales de 1966, Roberts entró en DARPA para desarrollar el concepto de redes informáticas y rápidamente creó su plan para "ARPANET", que publicó en 1967. En la conferencia en la que presentó el artículo había otra ponencia sobre el concepto de redes de paquetes, que venía del Reino Unido, de la

mano de Donald Davies y Roger Scantlebury, del NPL. Scantlebury le comentó a Roberts el trabajo del NPL y el de Paul Baran y otras personas de RAND. El grupo RAND había escrito un artículo sobre redes de conmutación de paquetes para cifrar comunicaciones de voz en el ejército en 1964. La labor del MIT (1961-1967), de RAND (1962-1965) y del NPL (1964-1967) se había llevado a cabo en paralelo sin que los investigadores conociesen el trabajo de los demás. Se adoptó el término "paquete" del trabajo del NPL, y la velocidad de línea propuesta en el diseño de ARPANET pasó de 2,4 kbps a 50 kbps. [5]

En agosto de 1968, después de que Roberts y la comunidad financiada por DARPA redefinieran la estructura general y las especificaciones de ARPANET, DARPA publicó una solicitud de presupuesto para desarrollar uno de los componentes clave, los conmutadores de paquetes llamados procesadores de mensajes de interfaz (IMP). La solicitud de presupuesto la ganó en diciembre de 1968 un grupo liderado por Frank Heart, de Bolt, Beranek y Newman (BBN). Mientras el equipo de BNN trabajaba en los IMP con Bob Kahn desempeñando un importante papel en el diseño arquitectónico general de ARPANET, Roberts, junto con Howard Frank y su equipo de Network Analysis Corporation, diseñaron la topología y la economía de la red. El sistema de medición de la red lo preparó el equipo de Kleinrock en UCLA.

Debido al temprano desarrollo de Kleinrock de la teoría de conmutación de paquetes y a su trabajo en el análisis, el diseño y la medición, su Network Measurement Center de UCLA fue seleccionado como el primer nodo de ARPANET. Se recogió el fruto de estos esfuerzos en septiembre de 1969, cuando BBN instaló el primer IMP en UCLA y se conectó el primer host. El proyecto de Doug Engelbart, "Augmentation of Human Intellect" (aumento del intelecto humano, que incluía NLS, un antecedente del sistema de hipertexto), en el Standford Research Institute (SRI), fue el segundo nodo. El SRI estaba detrás del Network Information Center, liderado por Elizabeth (Jake) Feinler, que incluía funciones como mantenimiento de tablas de nombres de host para asignar direcciones, así como de un directorio de RFC.

Un mes más tarde, cuando el SRI se conectó a ARPANET, se envió el primer mensaje de host a host desde el laboratorio de Kleinrock hasta el SRI. Se

añadieron dos nodos más, en la Universidad de California en Santa Bárbara y en la Universidad de Utah. Estos dos últimos nodos incorporaron proyectos de visualización de aplicaciones, con Glen Culler y Burton Fried, de la Universidad de California en Santa Bárbara, investigando métodos para mostrar funciones matemáticas usando pantallas de almacenamiento para resolver el problema de la actualización en la red, y Robert Taylor e Ivan Sutherland, de Utah, investigando métodos de representación 3D en la red. De esta manera, a finales de 1969, había cuatro hosts conectados en la ARPANET inicial, e Internet iniciaba su trayectoria. Incluso en esta primera etapa, conviene destacar que la investigación sobre redes incorporaba trabajo sobre la red subyacente y trabajo sobre cómo usar la red. Esta tradición continúa hoy en día.

En los siguientes años, se añadieron rápidamente ordenadores a ARPANET, y se siguió trabajando para conseguir un protocolo de host a host funcionalmente completo y otro software de red. En diciembre de 1970, el Network Working Group (NWG), bajo el liderazgo de S. Crocker, terminó el protocolo de host a host inicial de ARPANET, llamado Network Control Protocol (NCP). Cuando los sitios de ARPANET terminaron de implementar NCP, en el periodo de 1971 a 1972, los usuarios de la red pudieron, por fin, comenzar a desarrollar aplicaciones.

En octubre de 1972, Kahn organizó una gran demostración de ARPANET, que tuvo mucho éxito, en la International Computer Communication Conference (ICCC). Fue la primera demostración pública de esta nueva tecnología de redes. En 1972 también se introdujo la aplicación “hot” inicial, el correo electrónico. En marzo, Ray Tomlinson, de BBN, escribió el software básico de envío y lectura de mensajes de correo electrónico, motivado por la necesidad de los desarrolladores de ARPANET de un mecanismo sencillo de coordinación. En julio, Roberts amplió su utilidad escribiendo la primera utilidad de correo electrónico para hacer listas de mensajes, leerlos selectivamente, archivarlos, reenviarlos y responder a los mismos. A partir de ese momento, el correo electrónico se convirtió en la aplicación de red más importante durante más de una década. Esto presagió el tipo de actividad que vemos hoy en día en la World Wide Web, es decir, un

crecimiento enorme de todo tipo de tráfico “de persona a persona”. (Internet Society, 2012)

1.4 Metodología SCRUM

Scrum es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software. (Palacio, 2011, pág. 22)

1.4.1 Características

Es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el ScrumMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (interesados externos o internos), y el Team que incluye a los desarrolladores.
En Scrum se definen los llamados sprint el cual es un periodo entre que va de una y cuatro semanas (la magnitud es definida por el equipo), periodo en cual el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del ProductBacklog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar.
Los elementos del ProductBacklog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el ProductOwner identifica los elementos del ProductBacklog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint. Scrum permite la creación de equipos auto-organizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan (a

menudo llamado requirementschurn), y que estas nuevas definiciones impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido sobretodo en un inicio, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requerimientos emergentes. . (Palacio, 2011, pág. 30)

1.4.2 Definición de Sprints del proyecto

Para este proyecto se definirán sprints de 4 semanas cada uno en el cual se definirán las tareas a realizarse el equipo (en este caso los autores de la tesis).

Spring 1

En este spring se realizan todas las actividades relacionadas con la fase de análisis de requerimientos funcionales y no funcionales para el desarrollo del proyecto de software. Las tareas realizadas se detallan en la tabla 1.

Tabla 1

Sprint 1

Iteración	Tarea	Tipo	Responsable
H1	Análisis de los productos y giro de negocio de la empresa.	Análisis	Pablo Núñez
H2	Análisis de requisitos iníciales.	Análisis	Pablo Núñez
H3	Reconocimiento de implicados y requerimientos de sistema.	Análisis/Diseño	Pablo Núñez / Danny Quishpe
H4	Definición de roles de Usuario.	Análisis/Diseño	Pablo Núñez / Danny Quishpe
H5	Procesos del sistema y casos de Uso.	Análisis/Diseño	Pablo Núñez / Danny Quishpe

Elaborado por: Danny Quishpe, 2013

Spring 2

En el segundo spring se realizan las actividades relacionadas al diseño de software, es esta fase se selecciona la arquitectura del programa, la tecnología utilizada en el desarrollo de la aplicación y se diseña la base de datos. Un pequeño resumen de las tareas se presenta en la tabla 2.

Tabla 2

Spring 2

Iteración	Tarea	Tipo	Responsable
H1	Diseño de Arquitectura del sistema.	Diseño	Danny Quishpe
H2	Diseño de servicios web y protocolos de comunicación.	Análisis/Diseño	Danny Quishpe
H3	Selección de tecnologías de desarrollo y motor de base de datos.	Análisis	Pablo Núñez / Danny Quishpe
H4	Diseño de Base de datos.	Diseño	Pablo Núñez / Danny Quishpe
H5	Diccionario de datos.	Análisis	Pablo Núñez
H6	Diagramas de secuencia.	Diseño	Pablo Núñez
H7	Capas y diagrama de paquetes del sistema.	Diseño	Danny Quishpe
H8	Diagramas de Clase.	Diseño	Pablo Núñez / Danny Quishpe
H9	Diseño de Interfaces.	Diseño	Pablo Núñez

Elaborado por: Danny Quishpe, 2013

Spring 3

En el tercer spring se realizaron las tareas relacionadas a la construcción de la aplicación, basándose en el diseño realizado en la fase anterior se integró la aplicación web y móvil y se realizaron las conexiones necesarias para el manejo de la base de datos. Un detalle de las actividades realizadas se presenta en la tabla 3.

Tabla 3

Spring 3

Iteración	Tarea	Tipo	Responsable
H1	Conexión de base de datos y desarrollo (mapeo) del proyecto Core.	Construcción	Danny Quishpe
H2	Construcción de Servicio Web (Web Services).	Construcción	Danny Quishpe
H3	Construcción de Interfaces Web.	Construcción	Pablo Núñez
H4	Construcción de Interfaces Android.	Construcción	Danny Quishpe
H5	Pruebas del sistema	Pruebas	Pablo Núñez / Danny Quishpe

Elaborado por: Danny Quishpe, 2013

1.5 Lenguajes de programación Java con JSF

La tecnología Java Server Faces constituye un marco de trabajo (framework) de interfaces de usuario del lado de servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).

1.5.1 Características principales

Los principales componentes de la tecnología Java Server Faces son:

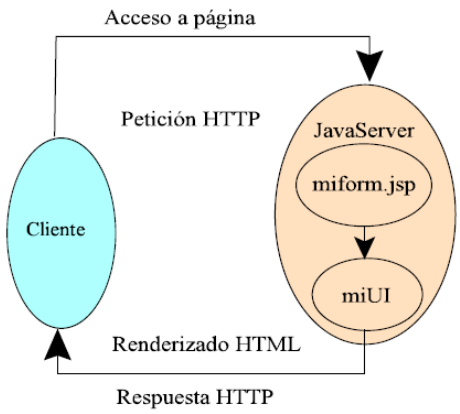
- Una API y una implementación de referencia para:
    - Representar componentes de interfaz de usuario (UI-User Interface) y manejar su estado.
    - Manejar eventos, validar en el lado del servidor y convertir datos.
    - Definir la navegación entre páginas.
    - Soportar internacionalización y accesibilidad, y proporcionar extensibilidad para todas estas características.
    - Una librería de etiquetas Java Server Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

Este modelo de programación bien definido y la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado servidor. Con un mínimo esfuerzo, es posible:

- Conectar eventos generados en el cliente a código de la aplicación en el lado servidor.
- Mapear componentes UI a una página de datos en el lado servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.

Como se puede apreciar en la siguiente figura, la interfaz de usuario que se crea con la tecnología Java Server Faces (representada por **miUI** en el gráfico) se ejecuta en el servidor y se renderiza en el cliente.

Figura 3 Ciclo de páginas JSF



Fuente: (Sistemas de Información Cooperativos de la Universidad de Málaga)

En la figura no se ve qué es físicamente **miUI**. La página JSP, **miform.jsp**, especifica los componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por la tecnología Java Server Faces. LA UI de la aplicación web (**miUI)** maneja los objetos referenciados por la JSP, que pueden ser de los siguientes tipos:

- Objetos componentes que mapean las etiquetas sobre la página JSP.
- Oyentes de eventos, validadores y conversores registrados y asociados a los componentes.
- Objetos del modelo que encapsulan los datos y las funcionalidades de los componentes específicos de la aplicación (lógica de negocio). (Sistemas de Información Cooperativos de la Universidad de Málaga)

¿Qué es una aplicación Java Server Faces?

En su mayoría, las aplicaciones Java Server Faces son como cualquier otra aplicación Web Java. Se ejecutan en un contenedor Servlet Java, y típicamente contienen:

- Componentes JavaBeans (llamados objetos del modelo en tecnología Java Server Faces) conteniendo datos y funcionalidades específicas de la aplicación.
- Oyentes de Eventos.
- Páginas, cómo páginas JSP.
- Clases de utilidad del lado del servidor, como beans para acceder a las bases de datos.

Además de estos ítems, una aplicación Java Server Faces también tiene:

- Una librería de etiquetas personalizadas para dibujar componentes UI en una página.
- Una librería de etiquetas personalizadas para representar manejadores de eventos, validadores, y otras acciones.
- Componentes UI representados como objetos con estado en el servidor.
- Validadores, manejadores de eventos y manejadores de navegación.

Toda aplicación Java Server Faces debe incluir una librería de etiquetas personalizadas que define las etiquetas que representan componentes UI y una librería de etiquetas para representar otras acciones importantes, como

validadores y manejadores de eventos. La implementación de Java Server Faces proporciona estas dos librerías.

La librería de etiquetas de componentes elimina la necesidad de codificar componentes UI en HTML u otro lenguaje de marcas, resultando en componentes completamente reutilizables. Y, la librería "core" hace fácil registrar eventos, validadores y otras acciones de los componentes.

La librería de etiquetas de componentes puede de la librería html_basic incluida con la implementación de referencia de la tecnología Java Server Faces, o podemos definir nuestra propia librería de etiquetas que dibuje componentes personalizados o que dibuje una salida distinta a HTML.

Otra ventaja importante de las aplicaciones Java Server Faces es que los componentes UI de la página están representados en el servidor como objetos con estado. Esto permite a la aplicación manipular el estado del componente y conectar los eventos generados por el cliente a código en el lado del servidor.

Finalmente, la tecnología Java Server Faces nos permite convertir y validar datos sobre componentes individuales y reportar cualquier error antes de que se actualicen los datos en el lado del servidor. (Torrijos, 2009)

1.6 Java para programación de aplicaciones para Android

¿Qué es Android?

Android es una plataforma de desarrollo libre, y de código abierto, el núcleo del sistema está basado en Linux 2.6 al que se le ha hecho ciertas modificaciones para que pueda ejecutarse en teléfonos y terminales móviles, que aunque cada vez son más potentes no dejan de tener menos recursos que un computador de escritorio. El hecho de ser gratuito y de código abierto, hace que los fabricantes de móviles puedan utilizarlo en sus nuevos terminales sin pagar licencias de uso.

Android cuenta con una gran cantidad de servicios disponibles, por ejemplo servicio de GPS incluidos mapas, lectores de código de barras o incluso bases de datos que servirá para mantener todos los datos de nuestra aplicación convenientemente ordenada y todo esto sin tener que instalar librerías externas ni configuraciones complejas como en otros entornos, los terminales también poseen una gran variedad de sensores que permiten tener conocimiento del entorno que los rodea y así poder acceder a la información para conocer la posición exacta del dispositivo, temperatura, etc.

1.6.1 Características y especificaciones actuales

1.6.2 Arquitectura

La arquitectura interna de la plataforma Android, está básicamente formada por cuatro componentes:

Aplicaciones: Todas las aplicaciones creadas con la plataforma Android, incluirán como base un cliente de email (correo electrónico), calendario, programa de SMS, mapas, navegador, contactos, y algunos otros servicios mínimos. Todas ellas escritas en el lenguaje de programación Java.

Framework de aplicaciones: Todos los desarrolladores de aplicaciones Android, tienen acceso total al código fuente usado en las aplicaciones base. Esto ha sido diseñado de esta forma, para que no se generen cientos de componentes de aplicaciones distintas, que respondan a la misma acción, dando la posibilidad de que los programas sean modificados o reemplazados por cualquier usuario sin tener que empezar a programar sus aplicaciones desde el principio.

Librerías: Android incluye en su base de datos un set de librerías C/C++, que son expuestas a todos los desarrolladores a través del framework de las aplicaciones Android System C library, librerías de medios, librerías de gráficos, 3D, SQlite, etc.

Runtime de android: Android incorpora un set de librerías que aportan la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. La Máquina Virtual está basada en registros, y corre clases compiladas por el compilador de Java que anteriormente han sido transformadas al **formato .dex** (Dalvik Executable) por la **herramienta ''dx''**. (MundoManuales, 2010)

CAPÍTULO 2

ANÁLISIS DE REQUERIMIENTOS FUNCIONES DEL SISTEMA

En este capítulo es el resultado de la ejecución del primer sprint que definió en la parte de la metodología en el capítulo uno.

2.1 Análisis de los productos y giro de negocio de la empresa

Rodamientos Bower es una empresa dedicada principalmente al comercio de rodamientos y retenedores, aunque existe una gran cantidad de empresas que se dedican a esta actividad económica una de las principales ventajas de esta empresa es su gran variedad de productos y marcas y un extenso stock lo cual le hace una de las empresas preferidas en Ambato.

Los automóviles y maquinaria pesada son los principales clientes de esta empresa, el primero por la simple razón que los rodamientos son uno de los principales repuestos que requiere un automotor sea de transporte pesado o un vehículo liviano. A esto se le suman algunas fábricas así como también parques de entretenimiento como por ejemplo Vulcano Park.

La gran cantidad de demanda que existe de este tipo de repuestas hace que la empresa tenga una gran salida y entrada de producto, tomando en consideración que una de las principales ventajas que se tendría en este mercado es la variedad de productos y un gran stock de productos, el control del inventario que se encuentra en bodega es fundamental, principalmente para evitar extracción no autorizada de productos y posible perdidas de ventas a gran estala por no tener un buen control del stock.

Un rodamiento es elemento mecánico que ayuda a reducir la fricción entre un eje y las piezas conectadas a este. En el mercado existen varios tipos de rodamientos entre los cuales tenemos rodamientos de rodillos o agujas, rodamientos radiales, rodamientos de bolas, etc. Así como tenemos varios tipos de rodamientos también existen varias marcas que los fabrican lo cual da como resultado una gran oferta de este producto.

2.2 Análisis de requisitos iníciales

El objetivo principal del sistema es el control del inventario de la empresa, debido a esto el programa deberá controlar tanto el ingreso como la salida de los productos de bodega de la empresa.

La empresa como tal no maneja solamente una marca de productos, por este motivo el sistema deberá ser capaz de manejar las diferentes marcas de productos que hay, así como también las diferentes clases o tipos de producto existentes. Los puntos a ser considerados en el sistema se detallan a continuación:

2.2.1 Manejo de Tipos de Productos

Para el sistema de control de inventario un tipo de producto puede ser un rodamientos especifico como por ejemplo un rodamiento de bolas, o simplemente se puede diferenciar entre rodamientos y retenedores que son las principales clases de producto que vende la empresa.

El nivel de detalle que tenga un tipo de producto dependerá directamente de la persona encargada de los mismos. Mientras más detallado sea un tipo de producto más fácil será la búsquedas de un producto específico en la tabla de productos.

2.2.2 Manejo de Marcas de Productos

La empresa como tal no se dedica a la fabricación de un producto sino al comercio de dicho producto, por lo cual en su bodega no se encuentran ítems de una marca específica, lo cual representa que el programa no solo deberá ser capaz de manejar diferentes tipos de productos sino que también deberá manejar varias marcas de productos.

2.2.3 Generación de Reportes

Una de las principales ayudas al usuario de un sistema de control de inventario son los reportes. El reporte más necesario en este caso es conocido como reporte de saldos rojos.

El reporte de saldos rojos consiste en todos los artículos que tengan un número de stock por debajo del número mínimo que debería existir en la bodega, este reporte ayuda principalmente a que siempre se encuentre un buen número de artículos en la bodega de una sucursal. Como las ventas en cada una de las sucursales es diferente el

número de stock aceptable también será diferente por lo que este número deberá ser manejado por sucursal.

Aunque no es parte de un sistema de control de inventario, el sistema a desarrollar también deberá presentar reportes del tipo gerencial, que muestren principalmente las ventas en un cierto lapso de tiempo, así como también productos y marcas preferidas por los clientes de la empresa. Este módulo se agregara principalmente para ayudar al departamento de contabilidad el cual no cuenta con un sistema contable que genere este tipo de reportes.

2.2.4 Manejo de Sucursales

Este proyecto será implementado en la oficina principal de la empresa, sin embargo el sistema debe ser capaz de manejar varias sucursales lo cual representa manejar varias bodegas con un solo sistema.

2.2.5 Auditoría y control de cambios

El sistema deberá guardar en las tablas para auditoría los datos del cualquier cambio que se realice en la tabla de stock de la empresa así mantener un control del inventario en bodega.

2.3 Reconocimiento de implicados y requerimientos de sistema

2.3.2 Lista de implicados

Las personas implicadas en este proyecto son empleados de la empresa ya que ellos serán los usuarios del software. Las personas implicadas por parte de la empresa en este proyecto se presentan en la tabla 4.

Tabla 4

Lista de Implicados

Nombre	**Cargo**	**Importancia**
Jorge Ramos	Gerente General	Alto
Cristian Ramos	Sub Gerente	Alto
Oswaldo Pérez	Jefe de Bodega	Alto
Mariana Romero	Auxiliar Contable	Medio

Elaborado por: Pablo Núñez, 2013

2.3.2 Requerimientos del software

2.3.2.1 Requerimientos Funcionales

- Para mantener un mejor control de las ventas y el stock, el software debe incluir un sistema de reportes donde se muestre la cantidad y el monto de los productos que se encuentran actualmente en bodega.
- Se debe generar un reporte de saldos rojo de los producto en bodega, con esto se evitara la falta de stock al momento de realizar una venta.
- El sistema debe ser diseñado para manejar lectura de código de barras.
- Para una mejor accesibilidad, el usuario debe ser capaz de acceder al software desde internet así como también desde dispositivos móviles.
- Las ventas y salida de productos será registrados en el software, y se podrá generar reportes del flujo de productos en bodega para una fecha específica.

2.3.2.2 Requerimientos no Funcionales

- El software debe ser desarrollado utilizando software libre para evitar gastos con respecto a pagos de licencias.
- El software deberá ser desarrollado utilizando una arquitectura orientada a servicios.
- El sistema móvil debe ser compatibles con varias versiones de Android.
- El sistema móvil deberá poder realizar procesos en segundo plano para evitar problemas en el uso normal de las demás aplicaciones del dispositivo.

2.4 Roles de Usuario

Para este proyecto se creará roles de acuerdo a las necesidades de cada uno de los departamentos de la empresa. En este casos se crearan solamente 3 roles.

- Rol Administrador
- Rol Bodega
- Rol Contabilidad
- Rol Ventas

El Administrador tendrá todos los permisos tanto para modificar información de los usuarios, así como también para ver todos los diferentes reportes que en software mostrara.

El rol de Bodega manejara todo lo referente a productos, tendrá todos los derechos para administrar tipos y marcas de productos así como también el ingreso de nuevo stock a la bodega.

El rol de Ventas tendrá acceso solamente a la pantalla de ventas de producto y será el encargado de registrar la salida de los productos hacia los clientes.

Por último el rol de Contabilidad solo tendrá acceso a información referente a las ventas que realice la empresa. En la tabla 5 se muestra cada proceso del sistema y que rol tiene acceso al mismo, si un rol tiene la capacidad de realizar una determinada acción tendrá marcado un signo X en la tabla.

Tabla 5

Permisos de acceso por roles de usuario

Rol de Usuario	**Administración de Usuarios**	**Ingreso de Productos**	**Venta o Salida de productos**	**Administración de Sucursales**	**Administración de Tipos de Producto**	**Administración de Marcas de Producto**	**Reportes**	**Datos de Auditoria**
Administrador	X	X	X	X	X	X	X	X
Bodega		X	X		X	X		
Contabilidad							X	
Ventas			X					

Elaborado por: Pablo Núñez, 2013

En caso de ser necesario nuevos roles estos no podrán ser ingresados directamente por el usuario y será necesario un cambio en la configuración del sistema.

2.5 Procesos del sistema y casos de Uso

2.5.1 Administración de Marcas de Producto

Esta funcionalidad permitirá el ingreso de marcas de los distintos productos al sistema así como también realizar alguna modificación o actualización luego de haber sido ingresadas, para luego poder usar estas marcas en el ingreso de los productos. Los datos del caso de uso de esta función se especifican en la tabla 6:

Tabla 6

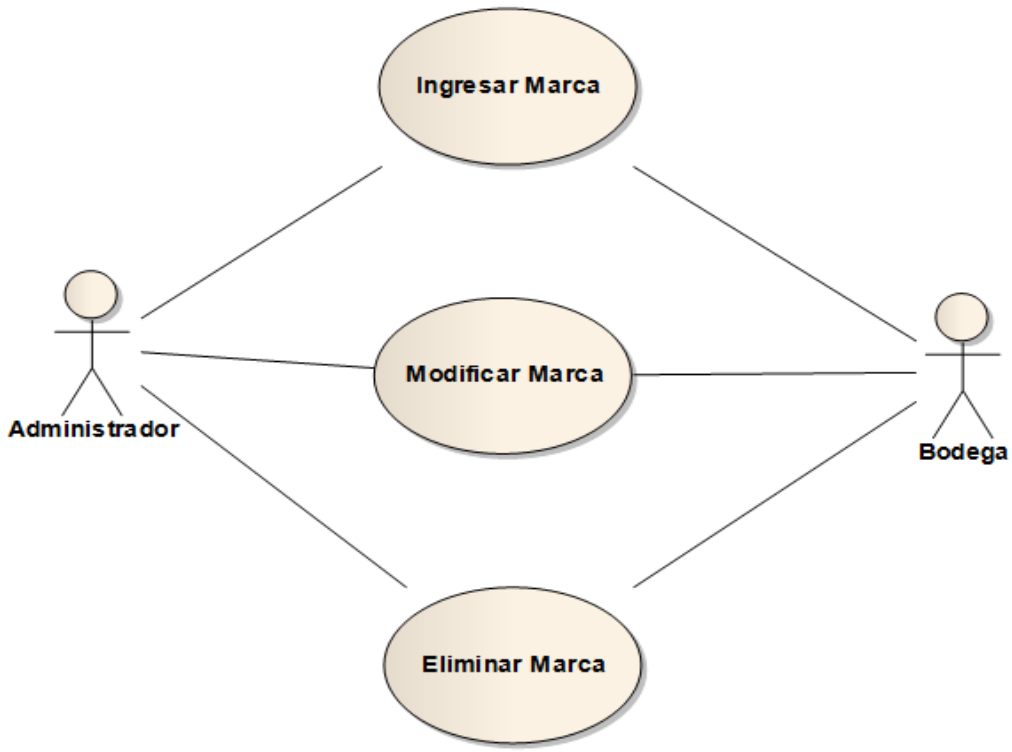
Caso de uso Administración de Marcas de Producto

Nombre	Administración de Marcas de Producto
Actores	Usuario Administrador Usuario de bodega
Precondiciones	El sistema debe estar levantado y funcionando Debe existir un rol de bodega o administrador activo disponible para que ejecute este funcionalidad
Poscondiciones	**Poscondiciones de éxito** Que los datos de la marca del producto se hayan ingresado correctamente listos para su utilización. Si se efectuó una actualización de marcas de productos que esta sea de manera correcta. **Poscondiciones de falla** Mensaje de error en la pantalla.
Escenarios	Este caso de uso inicia cuando se ingresa a la pantalla de Marcas de producto en la cual se puede realizar las siguientes acciones: - Ingresar una nueva marca de producto - Actualizar una marca existente **Flujo Básico** **Crear un nueva marca** Para agregar una nueva marca se debe dar clic en el botón Nuevo Marca, el cual abrirá una ventana emergente en el cual hay un formulario en el que se debe llenar los datos de la marca, los datos necesarios son: Nombre de la marca Descripción de la marca Una vez lleno el formulario se debe presionar el botón Guardar, una vez hecho esto en la pantalla principal aparecerá la marca ingresada además de todas las que se haya ingresado anteriormente. **Flujos Alternativos** **Actualizar marcas existentes** Para actualizar una marca previamente ingresada debemos dirigirnos a la tabla con las marcas de productos ingresados y dar clic en icono de actualizar en la fila de la marca a realizar los cambios, al hacer esto aparecerá una ventana emergente con todos los datos ya ingresados. En esta venta se podrá cambiar los datos deseados. **Fin del Caso de Uso** Termina el caso de uso.

Elaborado por: Pablo Núñez, 2013

Para una mejor visualización de las acciones que puede realizar cada uno de los usuarios en la figura 4 se presenta el diagrama del caso de uso.

Figura 4 Diagrama de Caso Uso Administración de Marcas de Producto



Elaborado por: Pablo Núñez, 2013

2.5.2 Administración de Sucursales

Esta funcionalidad permitirá el ingreso de sucursales al sistema así como también realizar alguna modificación de los datos de una sucursal ingresada anteriormente. Los datos del caso de uso y escenario de esta funcionalidad están detallados en la tabla 7.

Tabla 7

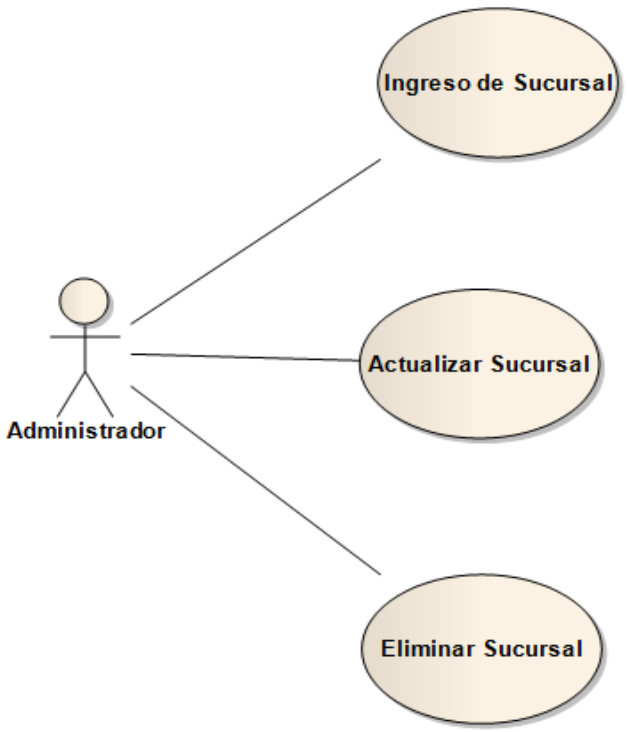
Caso de Uso Administración de Sucursales

Nombre	Administración de Sucursales
Actores	Usuario Administrador
Precondiciones	El sistema debe estar levantado y funcionando Debe existir un rol de administrador activo y disponible para que ejecute este funcionalidad
Poscondiciones	**Precondiciones de éxito** Que los datos de la sucursal hayan ingresado correctamente y se encuentren listos para su utilización. Si se efectuó una actualización de datos de la sucursal que esta sea de manera correcta. **Poscondiciones de falla** Mensaje de error en la pantalla.
Escenarios	Este caso de uso inicia cuando se ingresa a la pantalla de administración de sucursales en la cual se puede realizar las siguientes acciones: - Ingresar una nueva sucursal - Actualizar una sucursal existente **Flujo Básico** **Crear un nueva sucursal** Para agregar una nueva sucursal se debe dar clic en el botón Nueva Sucursal, el cual abrirá una ventana emergente en el cual hay un formulario en el que se debe llenar los datos de la sucursal, los datos necesarios son: Nombre de la sucursal Descripción de la sucursal Una vez lleno el formulario se debe presionar el botón Guardar, una vez hecho esto en la pantalla principal aparecerá la sucursal ingresada además de todas las que se haya ingresado anteriormente. **Flujos Alternativos** **Actualizar una sucursal existente** Para actualizar una sucursal previamente ingresada debemos dirigirnos a la tabla con las sucursales de productos ingresados y dar clic en icono de actualizar en la fila del sucursal a realizar los cambios, al hacer esto aparecerá una ventana emergente con todos los datos ya ingresados. En esta venta se podrá cambiar los datos que deseemos y luego procedemos a dar clic en el botón actualizar. **Fin del Caso de Uso** Termina el caso de uso.

Elaborado por: Pablo Núñez, 2013

Las acciones que puede realizar cada uno de los usuarios se representan en la siguiente figura.

Figura 5 Diagrama de Caso de uso Sucursales



Elaborado por: Pablo Núñez, 2013

2.5.2 Administración de Tipos de Producto

Esta funcionalidad permitirá el ingreso de tipos de productos al sistema así como realizar modificación o actualización luego de haber sido ingresadas. Los datos del caso de uso y escenario de esta funcionalidad están detallados en la siguiente tabla.

Tabla 8

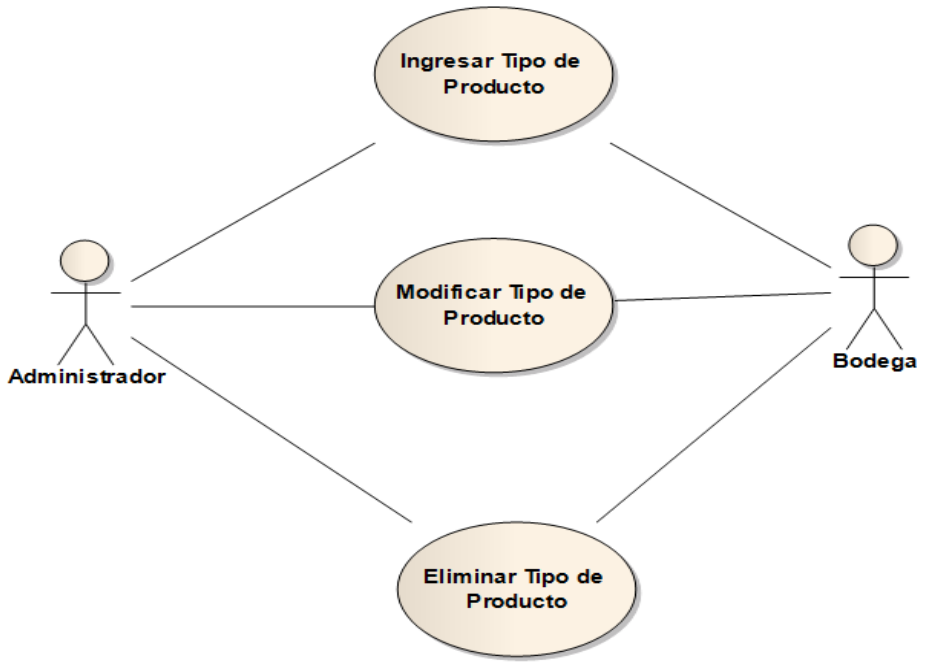
Caso de Uso Administración de Sucursales

Nombre	Administración de tipos de producto
Actores	Usuario Administrador
Precondiciones	El sistema debe estar levantado y funcionando Debe existir un rol de administrador activo y disponible para que ejecute esta funcionalidad
Poscondiciones	**Poscondiciones de éxito** Que los datos de tipo de producto se hayan ingresado correctamente y se encuentren listos para su utilización. Si se efectuó una actualización datos del tipo de producto que esta sea de manera correcta. **Poscondiciones de falla** Mensaje de error en la pantalla.
Escenarios	Este caso de uso inicia cuando se ingresa a la pantalla de administración de tipos de productos en la cual se puede realizar las siguientes acciones: - Crear un nuevo tipo de producto - Actualizar un tipo de producto existente **Flujo Básico** **Crear un nuevo tipo de producto** Para agregar un tipo de producto se debe dar clic en el botón **Nuevo Tipo de Producto**, el cual abrirá una ventana emergente en el cual hay un formulario en el que se debe llenar los datos del tipo de producto, los datos necesarios son: Nombre del tipo de producto Descripción del tipo de producto Una vez lleno el formulario se debe presionar el botón **Guardar**, una vez hecho esto en la pantalla principal aparecerá el tipo de producto ingresada además de todos los que se haya ingresado anteriormente. **Flujos Alternativos** **Actualizar un tipo de producto existente** Para actualizar un tipo de producto previamente ingresado debemos dirigirnos a la tabla con los tipos de productos ingresados y dar clic en icono de actualizar en la fila del tipo de producto a realizar los cambios, al hacer esto aparecerá una ventana emergente con todos los datos ya ingresados. En esta ventana se podrá cambiar los datos que deseemos y luego procedemos a dar clic en el botón actualizar. **Fin del Caso de Uso** Termina el caso de uso.

Elaborado: Pablo Núñez, 2013

Los actores que interactúan en este proceso y las diferentes acciones que pueden hacer se encuentran representados en la siguiente figura del diagrama de caso de uso.

Figura 6 Diagrama de Caso de Uso de Tipos de Productos



Elaborado por: Pablo Núñez, 2013

2.5.2 Administración de Usuarios

Esta funcionalidad permitirá el ingreso de nuevos usuarios al sistema, así como la actualización de los datos de cada usuario y activar o desactivar un usuario. Los datos de los escenarios y el caso de uso de este proceso se muestran en la tabla 10.

Tabla 9

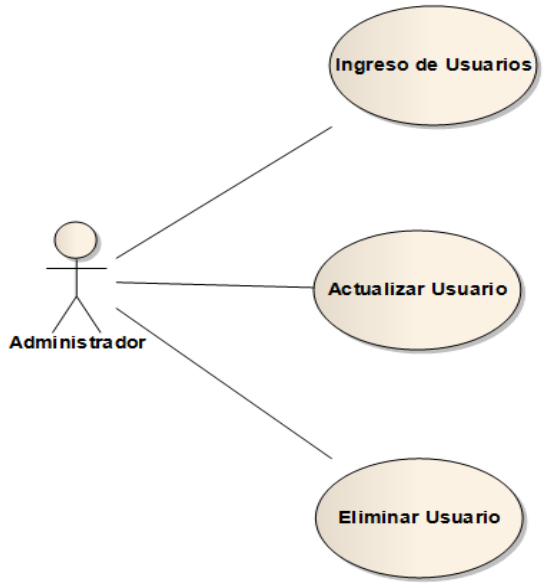
Caso de Uso Administración de Usuario

Nombre	Administración de usuarios
Actores	Usuario Administrador
Precondiciones	El sistema debe estar levantado y funcionando Debe existir un rol de administrador activo disponible para que ejecute este funcionalidad
Poscondiciones	**Poscondiciones de éxito** Si se efectuó un ingreso de usuario que este se haya creado correctamente y esté activo para su utilización. Si se efectuó una inactivación de usuario que este esté inhabilitado para poder ingresar y utilizar el sistema. **Poscondiciones de falla** Mensaje de error en la pantalla.
Escenarios	Este caso de uso inicia cuando se ingresa a la pantalla de Administración de Usuarios en la cual se puede realizar las siguientes acciones: - Crear un nuevo usuario - Actualizar un usuario existente **Flujo Básico** **Crear un nuevo usuario** Se deberá dar clic en el botón **Nuevo Usuario**, el cual abrirá una ventana emergente en el cual se deberán llenar los datos del nuevo usuario, los datos necesarios son: Identificación Nombre Apellido Rol de usuario Username Contraseña Confirmación de contraseña Activo/Inactivo Una vez lleno el formulario se debe presionar el botón **Guardar**, una vez hecho esto en la pantalla principal aparecerá el usuario ya ingresado en la tabla de usuarios. **Flujos Alternativos** **Actualizar un usuario existente** Para actualizar un usuario existente se debe dar clic en icono de actualizar en la fila del usuario a realizar los cambios, al hacer esto aparecerá una ventana emergente con todos los datos ya ingresados. En esta ventana se podrá cambiar los datos del usuario así como también desactivar o activarlo. **Fin del Caso de Uso** Termina el caso de uso.

Elaborado por: Pablo Núñez, 2013

En el siguiente diagrama se muestra las diferentes acciones que puede realizar el usuario administrador en esta pantalla:

Figura 7 Diagrama de Caso de Uso Administración de Usuarios



Elaborado por: Pablo Núñez, 2013

2.5.2 Administración de Productos

Esta funcionalidad permitirá el ingreso de productos al sistema así como realizar modificaciones o actualizaciones luego de haber sido ingresadas, para luego poder usar estos productos en el ingreso de stock de los mismos en las diferentes sucursales. Los escenarios y el caso de uso de este proceso se detallan en la tabla 10.

Tabla 10

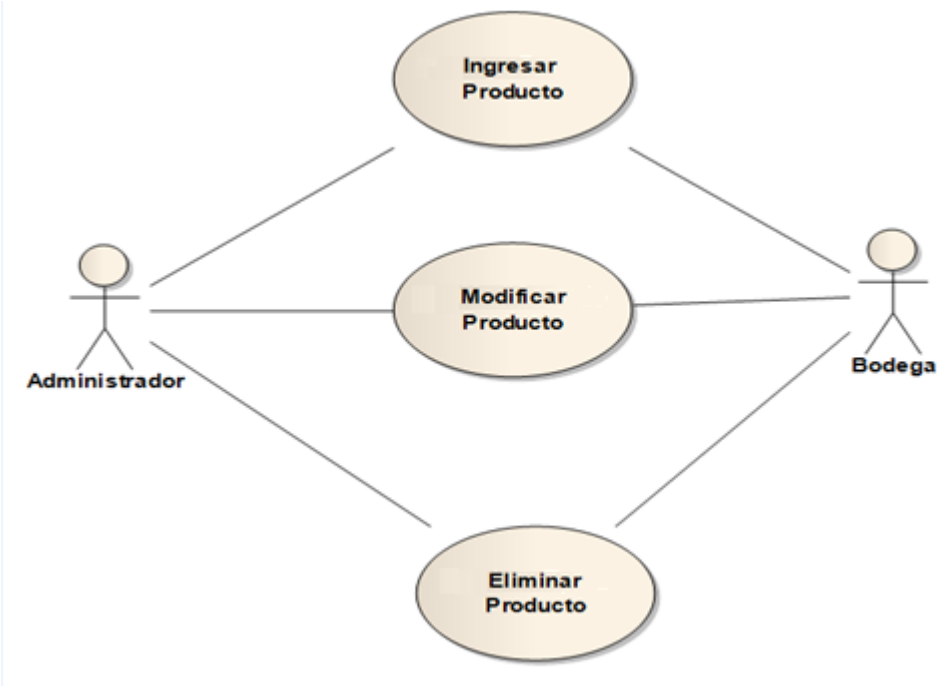
Caso de Uso Venta de Productos

Nombre	Ingreso de productos
Actores	Usuario Administrador Usuario de bodega
Precondiciones	El sistema debe estar levantado y funcionando Debe existir un rol de bodega o administrador activo disponible para que ejecute este funcionalidad Deben existir previamente tipos de productos y marcas de productos ya ingresados.
Poscondiciones	**Poscondiciones de éxito** Que los datos de los productos se hayan ingresado correctamente. Si se efectuó una actualización de productos que esta sea de manera correcta. **Poscondiciones de falla** Mensaje de error en la pantalla.
Escenarios	Este caso de uso inicia cuando se ingresa a la pantalla de producto en la cual se puede realizar las siguientes acciones: - Ingresar un nuevo producto - Actualizar un producto existente Flujo Básico **Crear un nuevo producto** Para agregar un nuevo producto se debe dar clic en el botón **Nuevo Producto**, el cual abrirá una ventana emergente en el cual hay un formulario en el que se debe llenar los datos de producto, los datos necesarios son: Nombre del producto Descripción del producto Tipo del producto Marca del producto Precio Stock mínimo Código de barras del producto Una vez lleno el formulario se debe presionar el botón **Guardar**, una vez hecho esto en la pantalla principal aparecerá la marca ingresada además de todas las que se haya ingresado anteriormente. Flujos Alternativos **Actualizar productos existentes** Para actualizar un producto previamente ingresado debemos dirigirnos a la tabla con los productos ingresados y dar clic en icono actualizar en la fila del producto a realizar los cambios, al hacer esto aparecerá una ventana emergente con todos los datos ya ingresados. En esta venta se podrá cambiar los datos que deseemos. **Fin del Caso de Uso** Termina el caso de uso.

Elaborado por: Pablo Núñez, 2013

En el siguiente diagrama se muestra las diferentes acciones que puede realizar el usuario administrador y de bodega en esta pantalla:

Figura 8 Diagrama de Caso de Uso Administración de Productos



Elaborado por: Pablo Núñez, 2013

Finalización del capítulo 2 y primer sprint.

CAPÍTULO 3

DISEÑO DEL SISTEMA

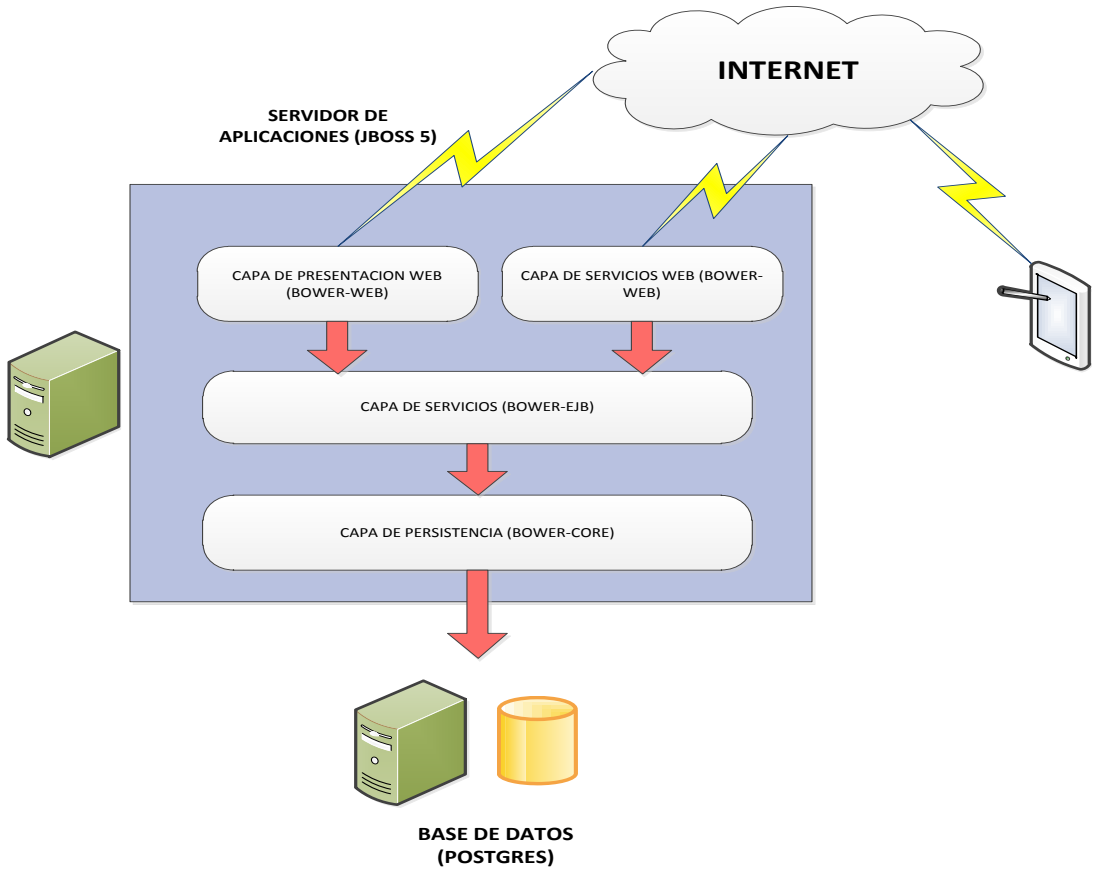
En este capítulo es el resultado de la ejecución del segundo sprint que definió en la parte de la metodología en el capítulo uno.

3.1 Diseño de Arquitectura del sistema

El software se manejara principalmente bajo un paradigma de una arquitectura orientada a servicios, los servicios que maneja el sitio web serán servicios Java, para el manejo de la aplicación móvil y para una mejor integración con otros lenguajes de desarrollo serán publicados servicios web.

La arquitectura que manejara el software se representa en la siguiente figura:

Figura 9 Arquitectura de Software del Proyecto Rodamientos Bower



Elaborado por: Danny Quishpe, 2013

Como se muestra en figura anterior tanto la página web como los servicios web serán desplegados en la nube o internet, por lo que la aplicación móvil podrá acceder a los datos desde cualquier lugar con acceso a internet, así como también la persona

encargada del manejo del sitio web podrá ingresar desde cualquier parte al sistema de control de inventario.

3.2 Diseño de servicios web y protocolos de comunicación

Los servicios web serán utilizados principalmente por la aplicación móvil que se desarrollara para Android, aunque los servicios web permiten intercambiar datos entre aplicaciones desarrollados en diferentes lenguajes para este caso se comunicaran dos aplicaciones desarrolladas en Java.

Los servicios web serán los encargados de brindar información con respecto a los stocks de productos. Como el aplicativo móvil tendrá la capacidad de leer códigos de barras será necesario un servicio que sea capaz de consultar el número del código de barras y retornar la información del producto consultado mediante la lectura de un código de barras.

La construcción de los servicios web se realiza utilizando el API de desarrollo JAX-WS, principalmente porque este API ayuda a simplificar el desarrollo y despliegue de los clientes y puntos finales de servicios web.

3.3 Selección de tecnologías de desarrollo y motor de base de datos

Uno de los requerimientos de este proyecto es que sea desarrollado con herramientas libres, en este caso se eligió uno de los lenguajes libres más conocimos y usados en el mundo, java. El sitio web será desarrollado utilizando el framework Java Sever Faces en su versión 2.0, el desarrollo de la aplicación se realizará utilizando más de un IDE de desarrollo sino dos Netbeans y Eclipse.

Netbeans será el IDE de desarrollo que se utilizará para programar el sitio web, mientras que eclipse será el IDE en el que se desarrollara la parte móvil en Android (Más información con respecto a JSF y programación para Android se encuentra en el capítulo uno).

Uno de los motores de base de datos más completos es PostgresSQL el cual es el que se utilizará en este proyecto, también se desarrollará el software utilizando tecnología hibérnate para el manejo de la data, lo cual permite mayor control con respecto a seguridad y manejo de los datos.

Las interfaces del sitio web se desarrollaran utilizando una de las librerías propias de JSF y primefaces, estas librerías ayuda a generar interfaces dinámicas de forma rápida y genera el código javascript necesario para la ejecución de las páginas en los distintos navegadores.

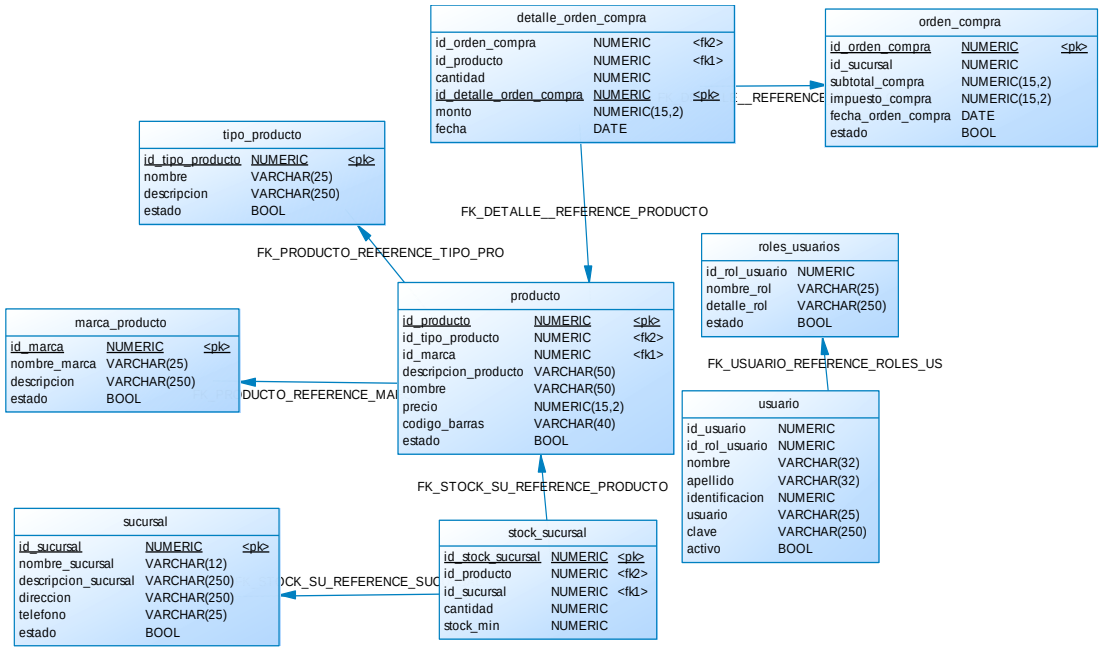
3.4 Diseño de Base de datos

3.4.1 Diseño físico de la base de datos

El diseño de la base de datos se pensó del tal forma que pueda cubrir todas las necesidades de un sistema de control de inventario. Para que el sistema pueda agregar otro módulos como uno de recursos humanos o un sistema contable será necesario realizar pocas modificaciones en el mismo ya que las tablas fueron diseñadas para que sea fácilmente adaptables y flexibles a cualquier modulo adicional.

A continuación se adjunta el diagrama del diseño físico de la base de datos del sistema.

Figura 10 Diagrama de diseño Físico de base de Datos

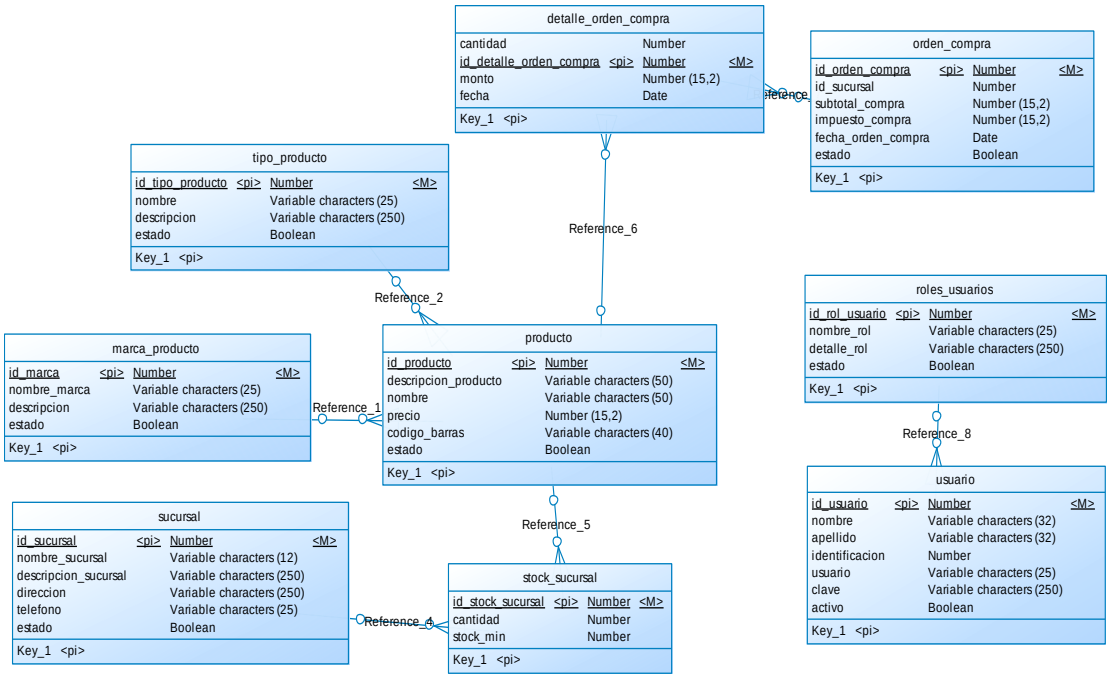


Elaborado por: Pablo Núñez, 2013

3.4.2 Diseño lógico de la base de datos

En la figura 11 se representa el diseño lógico de la base de datos.

Figura 11 Diagrama de Diseño Lógico de Base de Datos



Elaborado por: Pablo Núñez, 2013

3.5 Diccionario de datos

3.5.1 Secuencias de la base de datos

Tabla 11

Secuencias de Base de Datos

Nombre	**Descripción**	**Incremento**	**Inicio**
seq_detalle_orden_compra	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **detalle_orden_compra**	1	0
seq_marca_producto	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **marca_producto**	1	0
seq_orden_compra	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **orden_compra**	1	0
seq_producto	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **producto**	1	0
seq_roles	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **roles**	1	0
seq_sucursal	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **sucursal**	1	0
seq_tipo_producto	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **tipo_producto**	1	0
seq_usuarios	Esta secuencia es la encargada de generar los datos de la clave primaria de la tabla **usuario**	1	0

Elaborado por: Pablo Núñez, 2013

3.5.2 Tablas de la base de datos

Tabla detalle_orden_compra

En esta tabla se almacena información del detalle de cada compra que se realice, en esta tabla se guardara el código del producto almacenado, la cantidad vendida y el precio unitario del producto en ese instante (el precio puede variar en el tiempo por lo que es necesario almacenar el precio al instante de la venta).

Tabla 12

Columnas Tabla detalle_orden_compra

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_orden_compra	Integer	No	No	No	Si	No
id_producto	Integer	No	No	No	Si	No
Cantidad	Integer	No	No	No	No	No
id_detalle_orden_compra	Integer	nextval('seq_detalle_orden_compra'::regclass)	No	Si	No	Si
Monto	numeric(15,2)	No	No	No	No	No
Fecha	Date	getDate()	No	No	No	No

Elaborado por: Pablo Núñez, 2013

Tabla marca_producto

En esta tabla se encuentra información de los fabricantes de los diferentes productos que ofertan la empresa, esta información se utilizara para el ingreso de los productos.

Tabla 13

Columnas Tablas marca_producto

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_marca	Integer	nextval('seq_marca_producto'::regclass)	No	Si	No	si
nombre_marca	charactervarying(25)	No	No	No	No	no
Descripcion	charactervarying(250)	No	Si	No	No	no
Estado	Boolean	No	No	No	No	no

Elaborado por: Pablo Núñez, 2013

Tabla orden_compra

La tabla orden_compra contiene toda la información de las ventas realizadas en la empresa, esta información se encuentra clasificada según la sucursal donde se realizó

la compra así como también se encuentra información de la fecha de la compra el código del detalle de la misma.

Tabla 14

Columnas Tabla orden_compra

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_orden_compra	Integer	nextval('seq_orden_compra'::regclass)	No	Si	No	si
id_sucursal	Integer	No	No	No	Si	no
subtotal_compra	numeric(15,2)	No	No	No	No	no
impuesto_compra	numeric(15,2)	No	No	No	No	no
fecha_orden_compra	Date	No	No	No	No	no
Estado	Boolean	No	No	No	No	no

Elaborado por: Pablo Núñez, 2013

Tabla producto

La información de los diferentes productos que maneja la empresa se encuentra almacenada en esta tabla. Esta tabla contendrá el dato del precio actual del producto, así como el código de tipo de producto al que pertenece y el código de fabricante del mismo.

Tabla 15

Columnas Tabla producto

Nombre	Tipo de Dato	Valor porDefecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_producto	Integer	nextval('seq_producto'::regclass)	No	Si	No	si
id_tipo_producto	Integer	No	No	No	Si	no
id_marca	Integer	No	No	No	Si	no
descripcion_producto	charactervarying(150)	No	Si	No	No	no
Nombre	charactervarying(32)	No	No	No	No	no
Precio	numeric(15,2)	No	No	No	No	no
Estado	Boolean	No	No	No	No	no
codigo_barras	charactervarying(40)	No	No	No	No	no

Elaborado por: Pablo Núñez, 2013

Tabla roles_usuarios

La tabla de roles de usuario no puede ser administrado desde el sistema, la información contenida en esta tabla ayuda al software a validar que usuarios tienen acceso a cada una de las pantallas del mismo.

Tabla 16

Columnas Tabla roles_usuario

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_rol_usuario	Integer	nextval('seq_roles'::regclass)	No	Si	No	si
nombre_rol	charactervarying(25)	No	No	No	No	no
detalle_rol	charactervarying(250)	No	Si	No	No	no
Estado	Boolean	No	No	No	No	no

Elaborado por: Pablo Núñez, 2013

Tabla stock_sucursal

En esta tabla se guardaran los datos de la cantidad de productos que se encuentran en una sucursal determinada.

Tabla 17

Columnas Tabla stock_sucursales

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_stock_sucursal	integer	No	No	Si	No	Si
id_producto	integer	No	No	no	Si	No
id_sucursal	integer	No	No	no	Si	No
Cantidad	integer	No	Si	no	No	No
stock_min	integer	No	Si	no	No	No

Elaborado por: Pablo Núñez, 2013

Tabla sucursal

Esta tabla contiene la información de cada una de las sucursales que maneja el sistema.

Tabla 18

Columnas Tabla sucursales

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_sucursal	Integer	nextval('seq_sucursales'::regclass)	no	Si	No	si
nombre_sucursal	charactervarying(12)	No	no	No	No	si
descripcion_sucursal	charactervarying(250)	No	si	No	No	no
Direccion	charactervarying(250)	No	no	No	No	no
Telefono	charactervarying(25)	No	si	No	No	no
Estado	Boolean	No	no	No	No	no

Elaborado por: Pablo Núñez, 2013

Tabla tipo producto

La empresa comercializa diferentes tipos de productos, aunque la mayoría de productos son rodamientos también ofertan productos diferentes como retenedores, en la tabla de tipo de producto se almacena la información necesaria para poder clasificar a los productos de la empresa según su respectivo tipo.

Tabla 19

Columnas Tabla tipo_producto

Nombre	Tipo de Dato	Valor porDefecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_tipo_producto	Integer	nextval('seq_tipo_producto'::regclass)	No	Si	No	si
Nombre	charactervarying(25)	No	No	No	No	si
Descripción	charactervarying(250)	No	Si	No	No	no
Estado	Boolean	No	No	No	No	no

Elaborado por: Pablo Núñez, 2013

Tabla usuarios

En esta tabla se encuentra la información de los usuarios que manejaran el sistema, la clave del usuario se encuentran encriptados para mayor seguridad, ya que esta información se considera delicada. En esta tabla se asigna el código de rol de usuario.

Tabla 20

Columnas Tabla usuarios

Nombre	Tipo de Dato	Valor por Defecto	Permite Null	Clave primaria	Clave Foranea	Valor Unico?
id_usuario	Integer	nextval('seq_usuarios'::regclass)	no	Si	No	Si
id_rol_usuario	Integer	No	no	No	Si	No
Nombre	charactervarying (32)	No	no	No	No	No
Apellido	charactervarying (32)	No	no	No	No	No
Identificacion	Integer	No	no	No	No	No
Usuario	charactervarying (25)	No	no	No	No	si
Clave	charactervarying (250)	No	no	No	No	No
Active	Boolean	No	no	No	No	No

Elaborado por: Pablo Núñez, 2013

3.6 Diagramas de secuencia

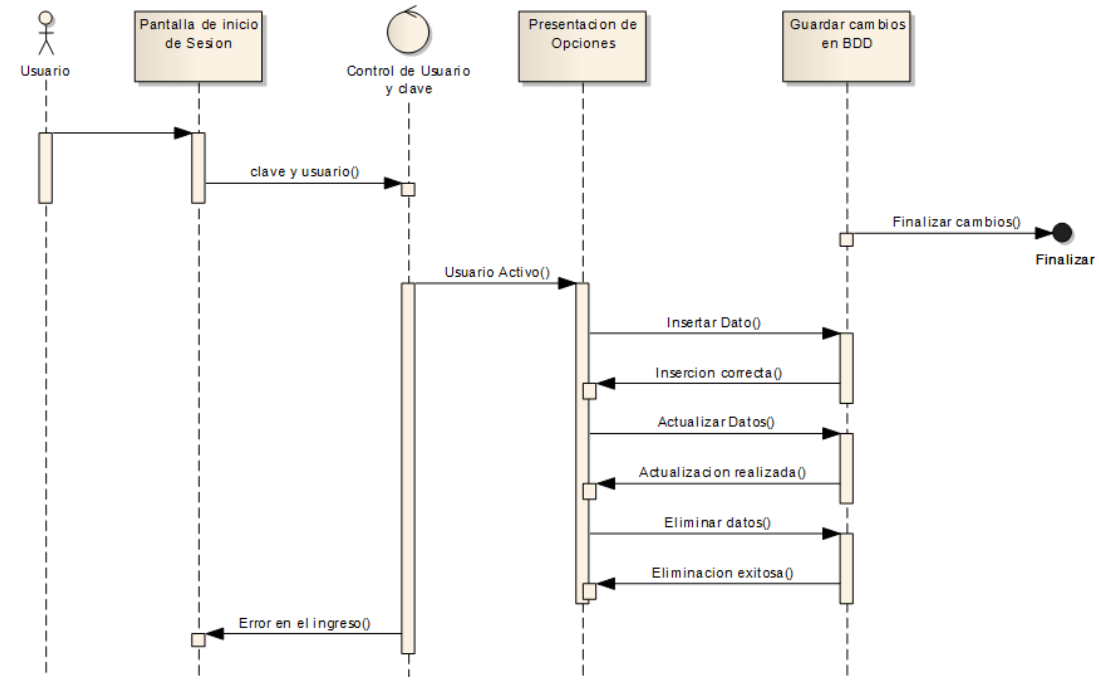
3.6.1 Diagrama de secuencia Pantallas de Administración

Se entiende por pantallas de administración a todas las pantallas que permiten insertar, modificar o eliminar un dato específico en la base de datos, sea que se refiera a un dato de un producto como al dato de una marca de producto.

Todas las pantallas de administración del sistema tiene un proceso de utilización muy simular una de otra, la diferencia entre cada pantalla son los tipos de datos que requiere cada una de las mismas para realizar un proceso especifico. La secuencia de cualquier pantalla de administración inicia con el ingreso del usuario en el sistema y finaliza cuando el usuario ha terminado los cambios.

Para una mejor representación de la secuencia de una pantalla de administración a continuación se presenta el diagrama de secuencias respectivo.

Figura 12 Diagrama de Secuencia de Pantallas de Administración



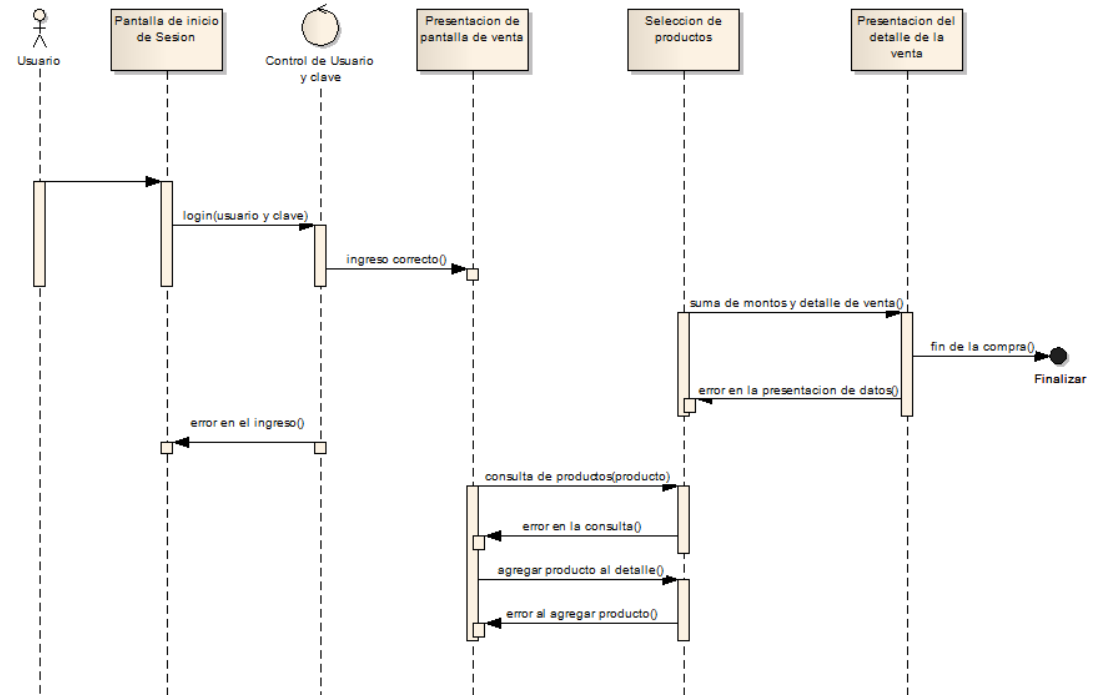
Elaborado por: Pablo Núñez, 2013

3.6.2 Diagrama de secuencia Pantallas de Venta de Productos

El proyecto está pensado para desarrollar un sistema de control de inventario, por lo que el módulo de venta de productos no estaría contemplado dentro del alcance del mismo. Sin embargo, por pedido de la empresa es necesario tener una pantalla que permita la salida de productos mediante la venta de los mismos.

La pantalla de venta de productos por ser este un sistema de control de inventario y no un sistema de ventas, el sistema no generara factura, pero al final del proceso de compra el sistema entregara un detalle de la orden de compra. El flujo se muestra en la figura 13 que se adjunta a continuación:

Figura 13 Diagrama de Secuencia Pantalla Venta de Productos

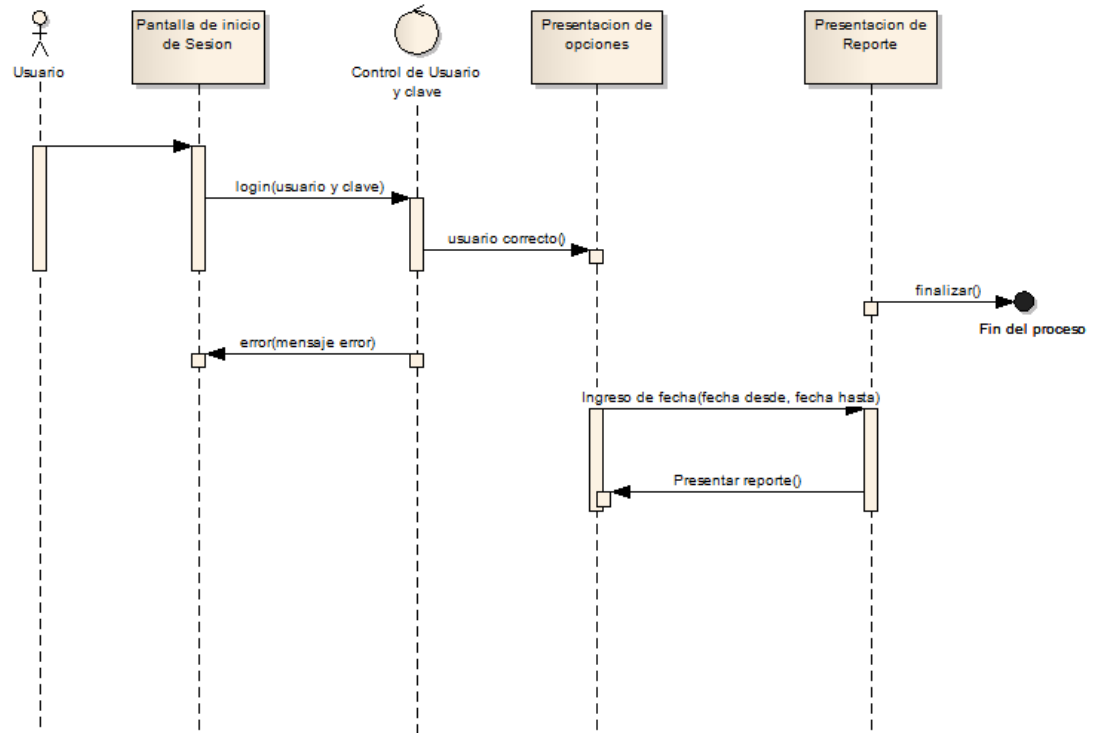


Elaborado por: Pablo Núñez, 2013

3.6.3 Diagrama de secuencia Pantallas de Reportes

Las pantalla de reportes servirán principalmente para tener un mejor control sobre el flujo y estado del inventario en bodega, el diseño de las pantallas permitirá que la información se pueda filtrar de acuerdo a las necesidades del usuario. Cada una de las pantallas de reportes tendrá sus propias opciones con respecto a filtros que se pueden aplicar en las mismas, sin embargo de esto el flujo de las pantallas será el mismo para todos los casos.

Figura 14 Diagrama de Secuencia de Pantalla de Reportes



Elaborado por: Pablo Núñez, 2013

3.7 Capas y diagrama de paquetes del sistema

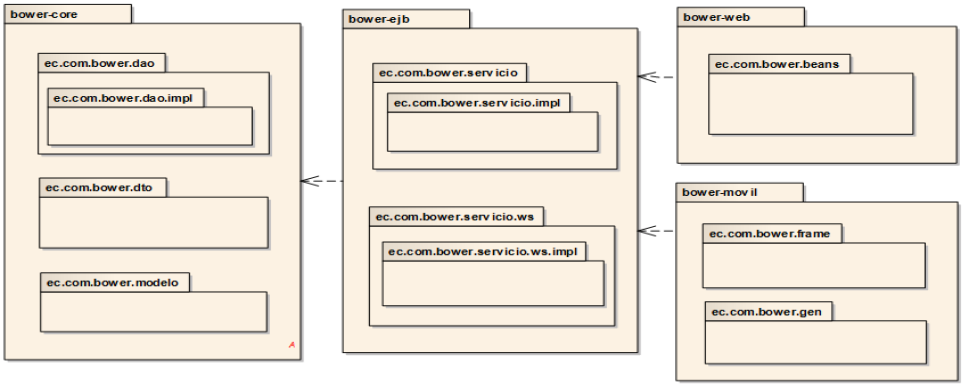
Para el desarrollo del sistema se va a seguir un diseño MVC (modelo, vista, controlador), por lo cual los paquetes del sistema se encuentran repartidos en estas tres capas lógicas.

En la capa de modelo se encontraran todos los paquetes necesarios para la conexión a base de datos y mapeo de la base, los paquetes que conforman el proyecto del bower-core son aquellos que se encuentran en la capa de modelo.

Para la capa de controlador tenemos el proyecto bower-ejb, en esta capa se encontraran las clases que realizan los procesos de negocio del sistema por ejemplo el proceso de guardado de datos, de verificación de identidad, etc.

Para finalizar se encuentran dos proyectos en la capa aplicación, el primero bower-web contiene todas las clases de la capa aplicación para el sitio web, mientras que el proyecto bower-movil contiene todas las clases para la aplicación móvil.

Figura 15 Diagrama de Paquetes del Sistema de Control de Inventario



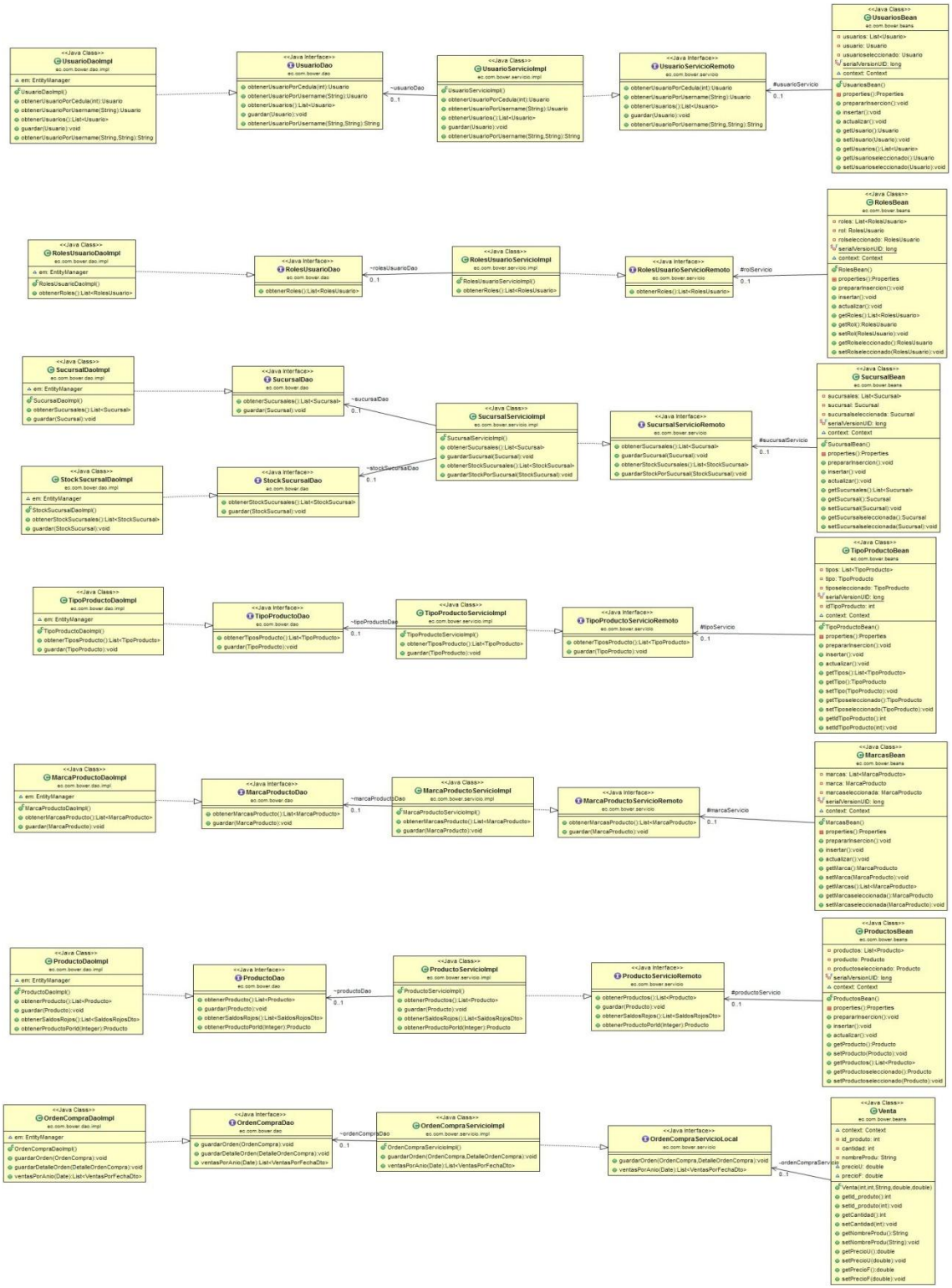
Elaborado por: Danny Quishpe, 2013

3.8 Diagramas de Clase

En los siguientes diagramas (UML) están representadas las clases que se usan dentro de la aplicación, en este se pueden ver como las clases están distribuidas en las distintas capas y como la comunicación de estas va desde las clases Dao (la construcción de estas clases se indica en el capítulo 4) hasta el nivel de las clases beans en la parte web. Las clases Dao cuentan con los métodos básicos de acceso a la base datos como lo son consultas y guardados, Las clases Service sirven como intermediarias entre la capa web y la de acceso a la base de datos además estas agrupan en sus métodos toda la lógica necesaria que es utilizada en la capa web. Las clases beans de la capa web usan toda la lógica de las clases service de la capa anterior además de constar con sus métodos y variables características para presentación de la capa web y lógica propia de cada pantalla.

En el siguiente diagrama se muestra la forma en que están estructuradas cada una de las clases además de cómo estas relacionan estas entre sí.

Figura 16 Diagrama de Clases del Sistema de Control de inventario



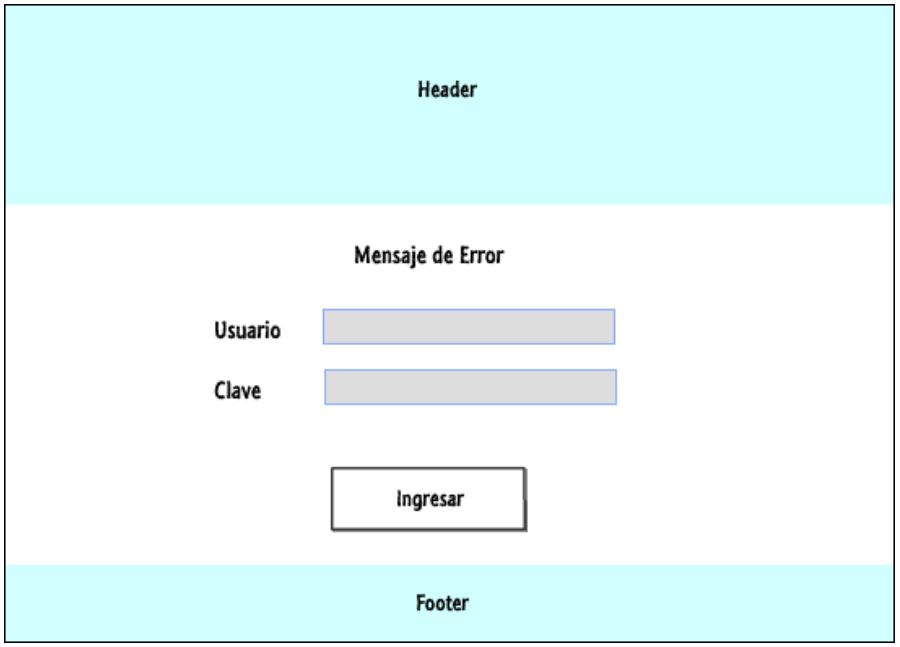
Elaborado por: Danny Quishpe, 2013

3.9 Diseño de Interfaces

3.9.1 Diseño de la página de inicio de sesión

La pantalla de login es la primera pantalla que se muestra en el sistema, los datos necesarios para el ingreso del usuario al programa son el nombre del usuario y la clave del usuario, los mismos que serán solicitados al momento del ingreso. El diseño de la pantalla de inicio se muestra en la figura 17.

Figura 17 Diseños de Interfaz Inicio de Sesión



Elaborado por: Pablo Núñez, 2013

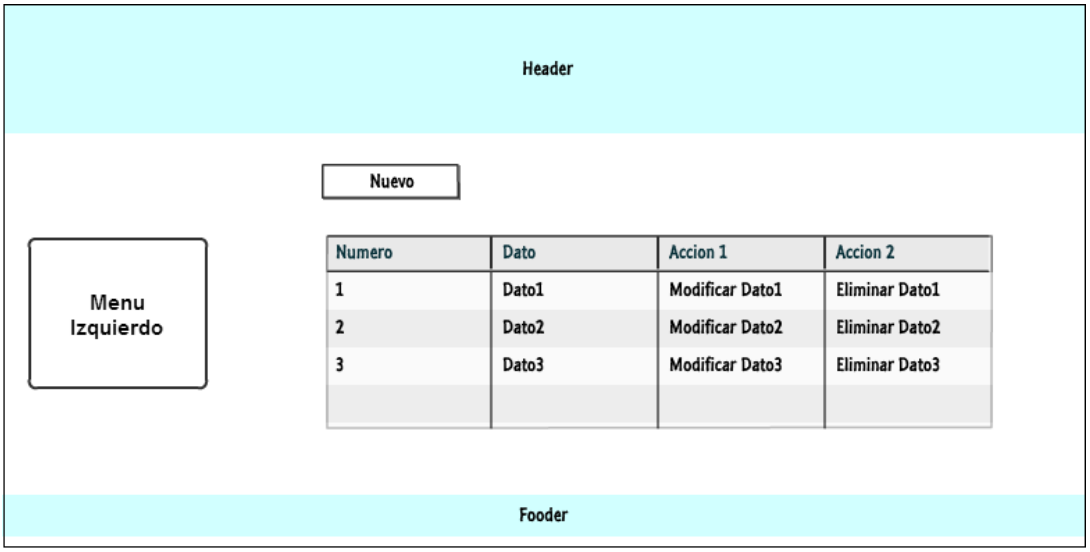
La pantalla de inicio de sesión tiene en la parte superior una cabecera con la información referente a la empresa como el nombre o un logo de la misma.

En la parte inferior se encuentra el pie de página con información de la ubicación de la empresa y de los derechos del programa. Para finalizar en la parte central se mostraran los campos necesarios para el inicio de sesión acompañados de botón para iniciar el proceso, en caso de existir algún problema en el ingreso un mensaje de error será desplegado en la parte superior de la pantalla.

3.9.2 Diseño de las pantallas de Administración

Las pantallas de administración como su nombre lo dice son las encargadas de modificar los datos de marcas, productos, tipos de producto, etc. También se puede ingresar nuevos datos en la base de datos a través de estas pantallas.

Figura 18 Diseño de Interfaces de Pantallas de Administración



Elaborado por: Pablo Núñez, 2013

En la parte superior de la página se encuentra la cabecera, que contiene el nombre de la empresa, el logo de la empresa e información de la página que servirá como guía para que el usuario.

Al costado izquierdo de la página se encontrara el menú del sistema de administración. La cantidad de opciones que se desplieguen en este menú dependerá del rol del usuario que inicie sesión.

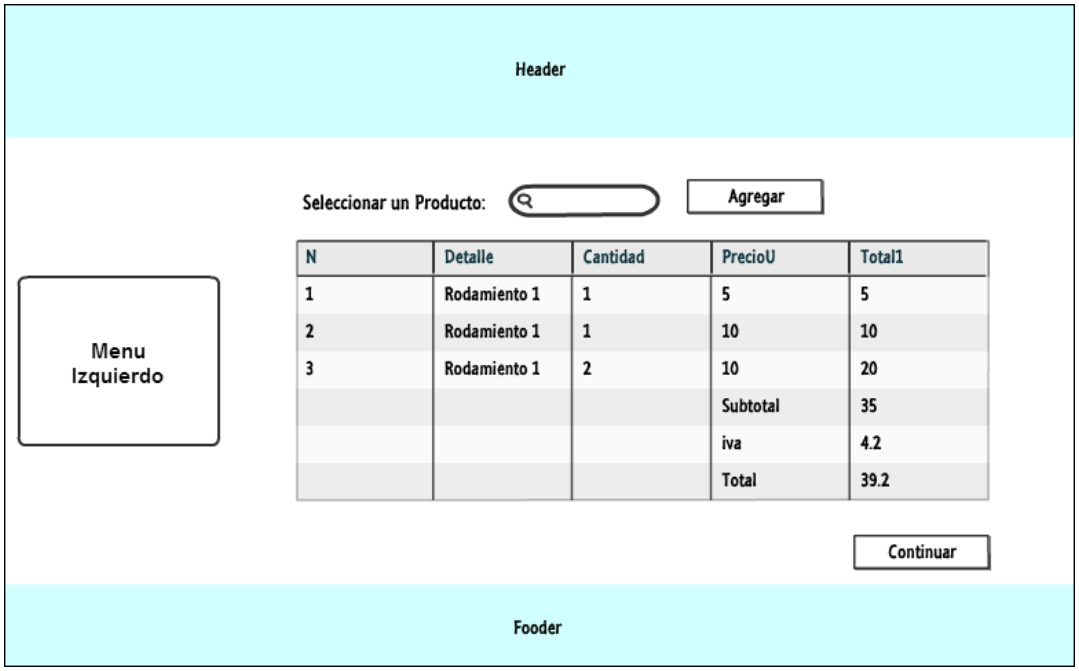
En la parte del medio a lado derecho del menú se encuentra un botón de nuevo, mediante este se ingresaran nuevos datos a la base de datos, debajo de este botón se podrá visualizar una tabla que contiene todos los datos previamente ingresados. La tabla contiene dos botones por cada fila de datos. El primer botón desplegara una pantalla de actualización de datos la cual visualmente es igual a la pantalla de ingreso de datos, con la diferencia de que esta pantalla tendrá ya cargado los datos de la fila seleccionada y actualizara el dato mas no lo ingresará.

Para finalizar en la parte inferior de la página web se encontrara el pie de página, en este se podrá visualizar información de la ubicación de la empresa e información referente al sistema de control de inventario que se desarrolla en este proyecto.

3.9.3 Diseño de las pantallas de Venta de productos

La compañía tiene como giro de negocio la venta de productos, el sistema de control de inventario necesita también un módulo para salida de productos o en este caso la venta de los mismo, en el diseño de estas pantallas se debe tomar en cuenta principalmente que se debe realizar una búsqueda de productos a vender y de un detalle de los productos que se va a vender. En el figura 19 se muestra el bosquejo de la pantalla de venta de productos.

Figura 19 Diseño Página de Ventas de Productos



Elaborado por: Pablo Núñez, 2013

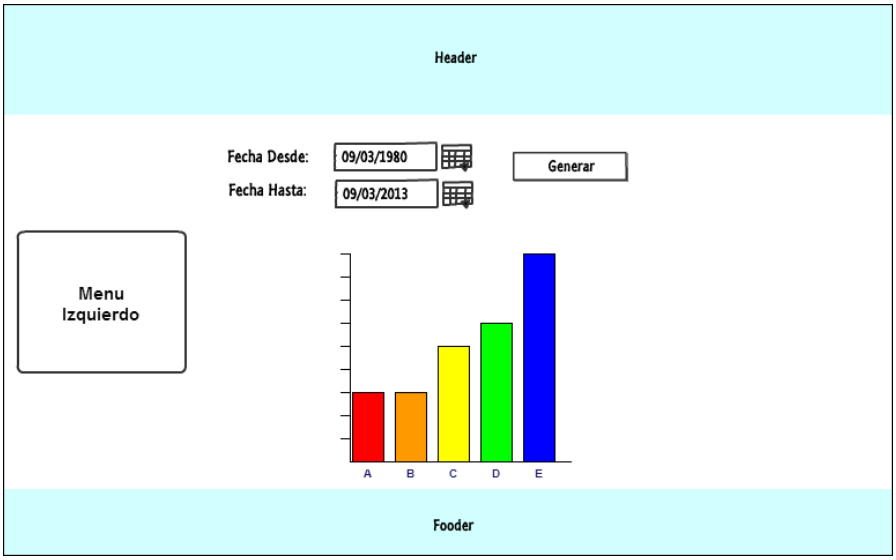
En el lado inferior izquierdo se muestra un detalle con todos los productos que se encuentran incluidos en la venta actual. En la parte inferior de la tabla se muestra la sumatoria de todos los totales de cada uno de los productos, también se incluirá información del I.V.A calculado y el total incluido el monto de los impuestos respectivos.

Muy cerca de la tabla de detalle en la parte superior se encuentra un buscador de productos. A lado del buscador se encuentra un botón que al momento de ser presionado agregara el producto buscado a la lista de productos del detalle de la compra. Aparte del botón para agregar un producto a la lista se encuentra en la parte inferior el botón continuar, este botón llevara a la pantalla de facturación de la compra con lo cual finalizaría el proceso.

3.9.4 Diseño pantallas de reportes

Las pantallas de reportes sirven principalmente como una herramienta de ayuda al gerente y al área de contabilidad de la empresa, por este motivo la gráfica presentada en la pantalla o la tabla de contenidos será la parte principal del diseño de la misma. A continuación en la figura 20 se muestra un bosquejo del diseño de las páginas de reportes.

Figura 20 Diseño Páginas de Reportes



Elaborado por: Pablo Núñez, 2013

Como se muestra en la ilustración al igual que en las pantallas de administración en el lado izquierdo de la pantalla se muestra el menú de acceso a todas las pantallas de administración.

En el diseño se muestra dos selectores de fechas, una vez elegido el rango de fechas del cual se quiere visualizar los datos se da clic en el botón generar y se generara un reporte en base a los datos solicitados en para ese rango de fechas.

CAPÍTULO 4

DESARROLLO DEL SISTEMA

Este capítulo es el resultado de la ejecución del tercer sprint que se definió en la parte de la metodología en el capítulo uno.

4.1 Conexión de base de datos y desarrollo (mapeo) del proyecto Core

Para el manejo de la base de datos se ha desarrollado un proyecto java denominado Core el cual está encargado de todo el manejo y la transacción con la base de datos construida en postgresql.

El sistema cuenta con una conexión a la base de datos hecha mediante JPA (Java Persistence API) El objetivo que persigue esta API es no perder las ventajas de la orientación a objetos al interactuar con la base de datos (siguiendo el patrón de mapeo objeto-relacional) y permitir usar objetos regulares (conocidos como POJO[1]s).

4.1.1 Definición de la unidad de persistencia

La unidad de persistencia define un conjunto de todas las entidades (clases) que son gestionadas por la instancia del EntityManager en la aplicación. Este conjunto de clases de entidad representa las tablas contenidas en la base de datos.

La unidad de persistencia del proyecto bower-core está definida en el fichero de configuración **persistence.xml**, el cual contiene el nombre del datasource[2] a utilizar además de declarar cada una de las clases o entidades correspondientes a las tablas de la base de datos.

[1]**POJO (Plain Old Java Object)**: Un objeto POJO es una instancia de una clase que no extiende ni implementa nada en especial.

[2]**Datasource:** representa todo lo relativo a una fuente de datos configurada por el usuario para conectarse a una Base de datos.

Tabla 21

Datos persistence.xml

persistence-unitname	**jta-data-source**	**Class**
bower-core	java:PostgresDS	ec.com.bower.modelo.DetalleOrdenCompra
		ec.com.bower.modelo.MarcaProducto
		ec.com.bower.modelo.OrdenCompra
		ec.com.bower.modelo.Producto
		ec.com.bower.modelo.RolesUsuario
		ec.com.bower.modelo.StockSucursal
		ec.com.bower.modelo.Sucursal
		ec.com.bower.modelo.TipoProducto

Elaborado por: Danny Quishpe, 2013

Otro fichero conocido como datasource también se debe configurar pero este se encuentra en el servidor de aplicaciones, el fichero llamado **postgres-ds**, en este fichero se debe especificar la dirección, nombre, usuario, contraseña y demás datos propios de la base de datos.

Tabla 22

Datos fichero postgres-ds

datasources	local-tx-datasource	jndi-name	**PostgresDS**	
		connection-url	**jdbc:postgresql:// 127.0.0.1:5432/bower**	
		driver-class	**org.postgresql.Driver**	
		user-name	**postgres**	
		password	**1234**	
		metadata	type-mapping	**PostgreSQL**

Elaborado por: Danny Quishpe, 2013

4.2 Construcción de entidades

Como se mencionó antes se debe definir un conjunto de entidades de persistencia las cuales vendrían a ser las tablas de la base de datos convertidas en clases java u objetos.

A continuación se presenta de una de estas entidades construidas a partir de una tabla de la base de datos.

Figura 21 Ejemplo de Entidad

```
@Entity
@NamedQueries({ @NamedQuery(name = "Sucursal.findAll", query = "select u from Sucursal u") })
@SequenceGenerator(name = "SucursalIdGenerator", sequenceName = "seq_sucursal",
allocationSize = 1)
@Table(name = "sucursal")
public class Sucursal implements Serializable {
        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(generator = "SucursalIdGenerator", strategy =
GenerationType.SEQUENCE)
        @Column(name = "id_sucursal")
        privateIntegeridSucursal;

        @Column(name = "descripcion_sucursal")
        private stringdescripcionSucursal;

        private string direccion;

        @Column(name = "nombre_sucursal")
        private string nombreSucursal;

        privateStringtelefono;

        public Sucursal() {
        }
        Public integer getIdSucursal() {
                returnthis.idSucursal;
        }
        publicvoidsetIdSucursal(IntegeridSucursal) {
                this.idSucursal = idSucursal;
        }
        publicStringgetDescripcionSucursal() {
                returnthis.descripcionSucursal;
        }
        publicvoidsetDescripcionSucursal(StringdescripcionSucursal) {
                this.descripcionSucursal = descripcionSucursal;
```

```
        }
        public string getDireccion() {
                returnthis.direccion;
        }
        public void setDireccion(Stringdireccion) {
                this.direccion = direccion;
        }
        public string getNombreSucursal() {
                returnthis.nombreSucursal;
        }
        public voids etNombreSucursal(string nombreSucursal) {
                this.nombreSucursal = nombreSucursal;
        }
        public string getTelefono() {
                returnthis.telefono;
        }
        public void setTelefono(String telefono) {
                this.telefono = telefono;
        }
}
```

Elaborado por: Danny Quishpe, 2013

Esta clase se encuentran anotaciones propias de JPA e Hibernate como el nombre de la tabla a la cual hace referencia, campos de la tabla, claves, secuencias, relaciones y querys a usarse, además de los métodos de acceso (getters y setters) característicos de un POJO.

4.3 Construcción de clases DAO (Data Access Object)

Las clases DAO son las que acceden a la base de datos a través del objeto EntityManager el cual es el que tiene las propiedades para realizar todas las operaciones de tipo CRUD (Selects, Inserts, Update, Deletes).

Figura 22 Clases Dao

```
@Stateless
public class SucursalDaoImpl implements SucursalDao {

	@PersistenceContext
	EntityManager em;

	@SuppressWarnings("unchecked")
	public List<Sucursal> obtenerSucursales() {

		Query query = em.createNamedQuery("Sucursal.findAll");
		return (List<Sucursal>) query.getResultList();
	}
	public void guardar(Sucursal sucursal) {

		if (sucursal.getIdSucursal() == null) {
			em.persist(sucursal);
		} else {
			em.merge(sucursal);
		}
	}
}
```

Elaborado por: Danny Quishpe, 2013

En estas clases DAO se definen todas las operaciones que se utilizarán en la aplicación ya sean consultas, ingresos o actualizaciones de datos. Para guardar una estructura estándar se creara una clase DAO por cada entidad de la base datos.

4.4 Construcción de Servicio Web (Web Services)

Para la construcción de estos servicios web utilizamos el API de desarrollo de java JAX-WS el cual provee anotaciones sencillas para construir el servicio web.

La construcción de un servicio web es sencilla puesto que lo único que se debe hacer es crear una clase JAVA común y crear los métodos necesarios para la funcionalidad

a exponer como servicio web. Lo que realmente convierte en servicio web a una clase es la interfaz de la cual se implementa, ya que aquí es donde usan las anotaciones necesarias para que sea expuesta como servicio web en el servidor de aplicaciones.

Figura 23 Interface de servicio web

```
@WebService
@SOAPBinding(style=SOAPBinding.Style.RPC)
public interface ConsultasServiciosWs extends Remote{

        @WebMethod
        @WebResult(name = "productoDto")
        public ProductoDto obtenerProductoPorIdSucursal(
                        @WebParam(name = "idProducto") Integer idProducto,
                        @WebParam(name = "idSucursal") Integer idSucursal);
}
```

Elaborado por: Danny Quishpe, 2013

Las anotaciones utilizadas son las siguientes

@WebService: sirve para definir como servicio web a la clase.

@SOAPBinding: definimos al servicio web como tipo SOAP

@WebMethod: define como método web a las operaciones a exponer en el servicio web.

@WebResult(name = "productoDto"): define el nombre del resultado que va a retornar el servicio web, en este caso un objeto productoDto.

@WebParam(name = "idProducto"): define el nombre de un parámetro que va a recibir método web que va a ser expuesto.

Una vez que el servicio web sea expuesto en el servidor de aplicaciones se podrá verificar el uso de anotaciones en la estructura del WSDL.

Figura 24 Fichero wsdl del servicio web

```
<?xmlversion="1.0"encoding="UTF-8"?>
<definitionsname='ConsultasServiciosWsImplService'targetNamespace='http://ws.servicio.bower.
com.ec/'xmlns='http://schemas.xmlsoap.org/wsdl/'xmlns:ns1='http://ws.servicio.bower.com.ec/'x
```

```
mlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'xmlns:tns='http://impl.ws.servicio.bower.com.
ec/'xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
<types>
<xs:schematargetNamespace='http://ws.servicio.bower.com.ec/'version='1.0'xmlns:xs='http://www
.w3.org/2001/XMLSchema'>
<xs:complexTypename='productoDto'>
<xs:sequence>
<xs:elementminOccurs='0'name='idProducto'type='xs:int'/>
<xs:elementminOccurs='0'name='marcaProducto'type='xs:string'/>
<xs:elementminOccurs='0'name='nombreProducto'type='xs:string'/>
<xs:elementminOccurs='0'name='precio'type='xs:double'/>
<xs:elementminOccurs='0'name='stock'type='xs:int'/>
<xs:elementminOccurs='0'name='sucursal'type='xs:string'/>
<xs:elementminOccurs='0'name='tipoProducto'type='xs:string'/>
</xs:sequence>
</xs:complexType>
</xs:schema>
</types>
<messagename='ConsultasServiciosWs_obtenerProductoPorIdSucursal'>
<partname='idProducto'type='xsd:int'/>
<partname='idSucursal'type='xsd:int'/>
</message>
<messagename='ConsultasServiciosWs_obtenerProductoPorIdSucursalResponse'>
<partname='productoDto'type='ns1:productoDto'/>
</message>
<portTypename='ConsultasServiciosWs'>
<operationname='obtenerProductoPorIdSucursal'parameterOrder='idProductoidSucursal'>
<inputmessage='ns1:ConsultasServiciosWs_obtenerProductoPorIdSucursal'/>
<outputmessage='ns1:ConsultasServiciosWs_obtenerProductoPorIdSucursalResponse'/>
</operation>
</portType>
<bindingname='ConsultasServiciosWsBinding'type='ns1:ConsultasServiciosWs'>
<soap:bindingstyle='rpc'transport='http://schemas.xmlsoap.org/soap/http'/>
<operationname='obtenerProductoPorIdSucursal'>
<soap:operationsoapAction=''/>
<input>
<soap:bodynamespace='http://impl.ws.servicio.bower.com.ec/'use='literal'/>
</input>
<output>
<soap:bodynamespace='http://impl.ws.servicio.bower.com.ec/'use='literal'/>
```

```
</output>
</operation>
</binding>
</definitions>
```

Elaborado por: Danny Quishpe, 2013

4.5 Construcción de Interfaces Web

Las interfaces web están desarrolladas usando JSF por lo que su funcionamiento se divide en varias capas, la capa de presentación o frontEnd utiliza PrimeFaces para el manejo del javascript y el control de los diferentes eventos que sea necesario controle la página.

Cada una de las páginas se encuentra compuesta de dos partes principales la páginaxhtml y el bean que controla la página.

4.6 Creación de archivos xhtml

La página xhtml es casi igual a una página html con la diferencia de que esta página maneja etiquetas JSP para el manejo de los objetos web. En esta página se encontraran las referencias de archivos JS y CSS tal y como se maneja en html normal.

Figura 25 Interface de Login

Elaborado por: Pablo Núñez, Octubre 2013

En la figura 25 se muestra una captura de la pantalla de inicio del programa. Como se puede observar los textfield de ingreso de usuario y clave se encuentra generado con un diseño que maneja la librería primefaces, el mismo caso lo podemos observar en el diseño del botón de ingreso al sistema.

4.7 Generación de Beans y manejo de datos

El control de datos se realiza por medio de los beans, un bean puede ser de diferente tipo, desde un bean de sesión como también un bean de administración, para explicar de mejor están formados los beans del proyecto tomaremos como referencia la página de inicio.

La página para ingresar al sistema requiere del ingreso de dos parámetros el usuario y la clave, como el bean manejará estos datos es necesario crear dos variables para el manejo de los mismos. Estas variables deberán estar instanciadas y generados los geters y seters respectivos para que puedan ser llamados desde el lado xhtml. Por ejemplo en el caso del parámetro usuario tenemos generada la variable del tipo String nombrada user, para llamar a la referencia de esta variable simplemente se debe poner el nombre de la misma en la etiqueta jsp tal y como se maneja objetos.

```
<p:inputTextvalue="#{loginUsuario.user}" rendered=true />
```

La etiqueta p fue designada para la librería primefaces. El bean de manejo de xhtml se comunica con la base de datos por medio de los servicios java, para eso es necesario instanciar la variable del servicio java, por ejemplo para la verificación del ingreso de usuario y clave correctos en la llamada a los servicios java se realizó así:

Figura 26 Ejemplo código context

```
String usuarioBuscado;
        context = new InitialContext(properties());
        Objectobject = context.lookup("UsuarioServicioImpl/remote");
        usuarioServicio = (UsuarioServicioRemoto) PortableRemoteObject.narrow(object,
UsuarioServicioRemoto.class);
        usuarioBuscado = usuarioServicio.obtenerUsuarioPorUsername(this.pass, this.user);
```

Elaborado por: Danny Quishpe, 2013

Para la verificación se utiliza el servicio Usuario el mismo que se utiliza en la pantalla de administración de usuario, para que este servicio funcione correctamente es necesario también llamar al método properties el cual inicializa los datos necesarios para el mapeo de objetos.

Figura 27 Ejemplo Método properties

```
private Properties properties() {
        Properties properties = new Properties();
        properties.put(Context.INITIAL_CONTEXT_FACTORY,
        "org.jnp.interfaces.NamingContextFactory");
        properties.put(Context.URL_PKG_PREFIXES,
        "org.jboss.naming:org.jnp.interfaces");
        properties.put(Context.PROVIDER_URL, "jnp://localhost:1099");
        return properties;
}
```

Elaborado por: Danny Quishpe, 2013

Después que los datos del usuario fueron comprobados la página procederá a crear las variables de sesión correspondiente para el control del ingreso del mismo y re direccionar a la siguiente página del flujo.

Cada bean fue diseñado y desarrollado acorde a las necesidades de la página que va a manejar, también fueron desarrollados los métodos necesarios para que el bean realice diferentes procesos de comprobación y funcionalidad. Todos los beans se encuentran en el paquete ec.com.bower.beans.

Para el caso de las pantallas de administración primero son generados listas de objetos del bean. Estos objetos se cargan en el grid que presenta los datos de la entidad respectiva. La figura 27 muestra la pantalla de administración de usuarios del sistema.

Figura 28 Pantalla de Administración de Usuarios

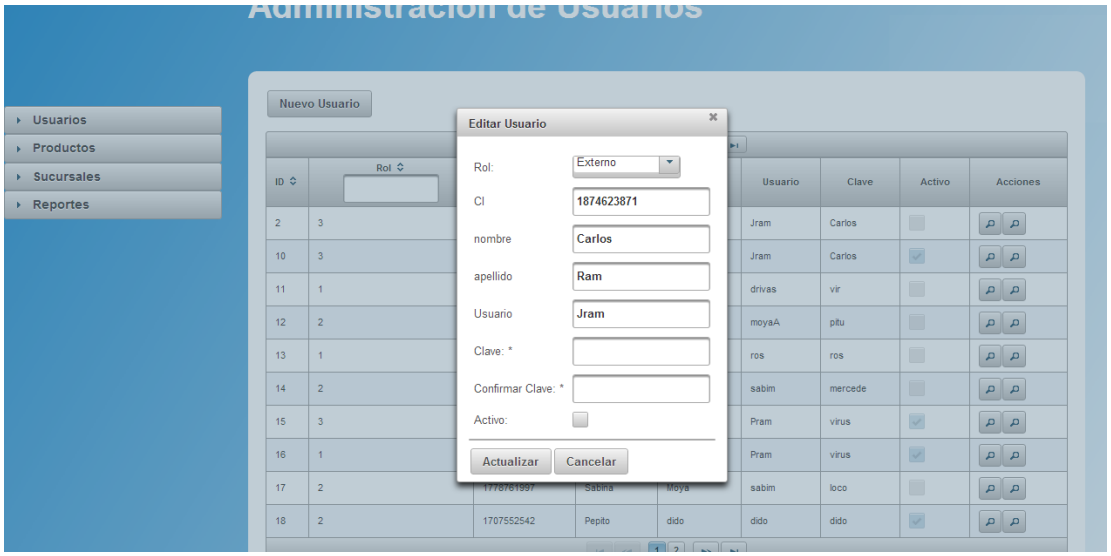
Elaborado por: Pablo Núñez, 2013

Como se puede observar en la figura 28 la pantalla contiene un botón para la creación de un nuevo usuario en la parte superior y botones para actualizar un usuario los cuales se encuentra en la columna de acciones dentro del grid.

Para el caso de creación de un nuevo usuario, una vez presionado el botón de Nuevo Usuario se despliega un pantalla desplegable con todos los campos necesarios para el ingreso de un nuevo Usuario, una vez finalizado el proceso de ingreso de datos al presionar el botón guardar se cerrara la pantalla desplegable y se puede visualizar que el nuevo usuario ingresado ya se encuentra en el grid de usuarios.

El mismo caso se presenta para la columna de actualización de datos, donde a diferencia del caso de nuevo usuario la pantalla desplegable ya tiene precargada los datos del usuario seleccionado y en el grid de datos de usuarios se muestra la fila actualizada. La figura 29 muestra la pantalla de actualización de datos de un usuario.

Figura 29 Pantalla de Actualización de Usuarios



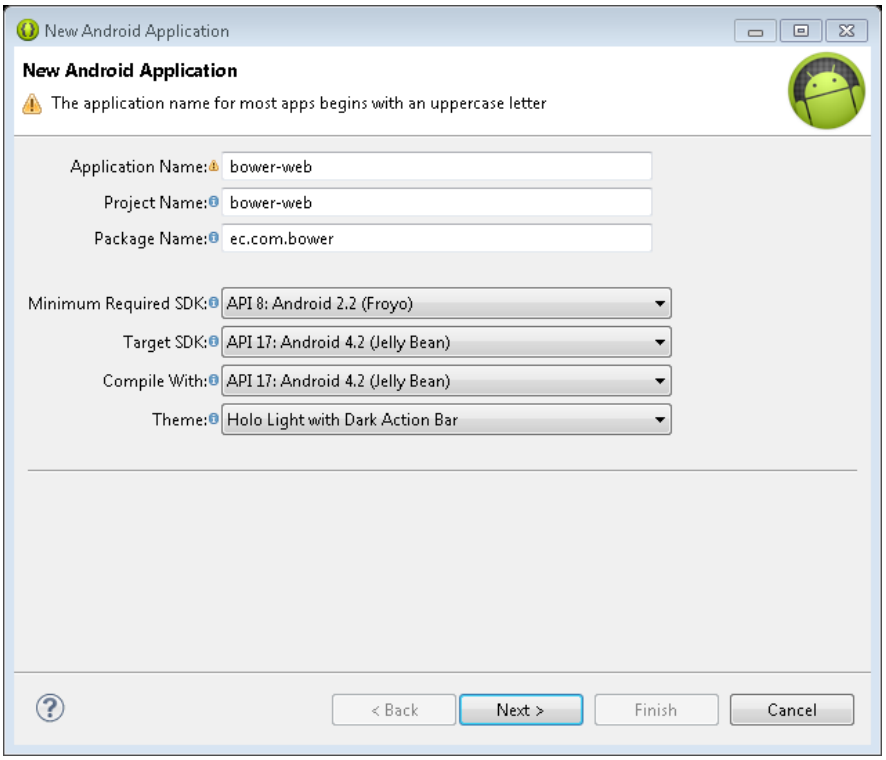
Elaborado por: Pablo Núñez, 2013

4.8 Construcción de Interfaces Android

Para la construcción de la aplicación móvil en android y por ende sus interfaces es necesario contar con el ambiente de trabajo listo es decir tener listas y configuradas todas las herramientas necesarias las cuales son básicamente el Kit de desarrollo android que consta de: SDK con todas la librerías android, la máquina virtual de andriod la cual sirve para hacer pruebas y el IDE de desarrollo eclipse configurado con todos los plugins y herramientas para desarrollo android.

Una vez listo el ambiente se procede a crear un proyecto android en el IDE de desarrollo, al crear el proyecto de debe especificar características como: nombre de la aplicación, nombre de proyecto, versiones de android a utilizar, etc.

Figura 30 Creación de las Pantallas de Interfaz de Usuario



Elaborado por: Danny Quishpe, 2013

La interfaz de usuario o pantalla se define en los archivos XML del directorio res/layout. Cada pantalla tendrá un fichero XML diferente.

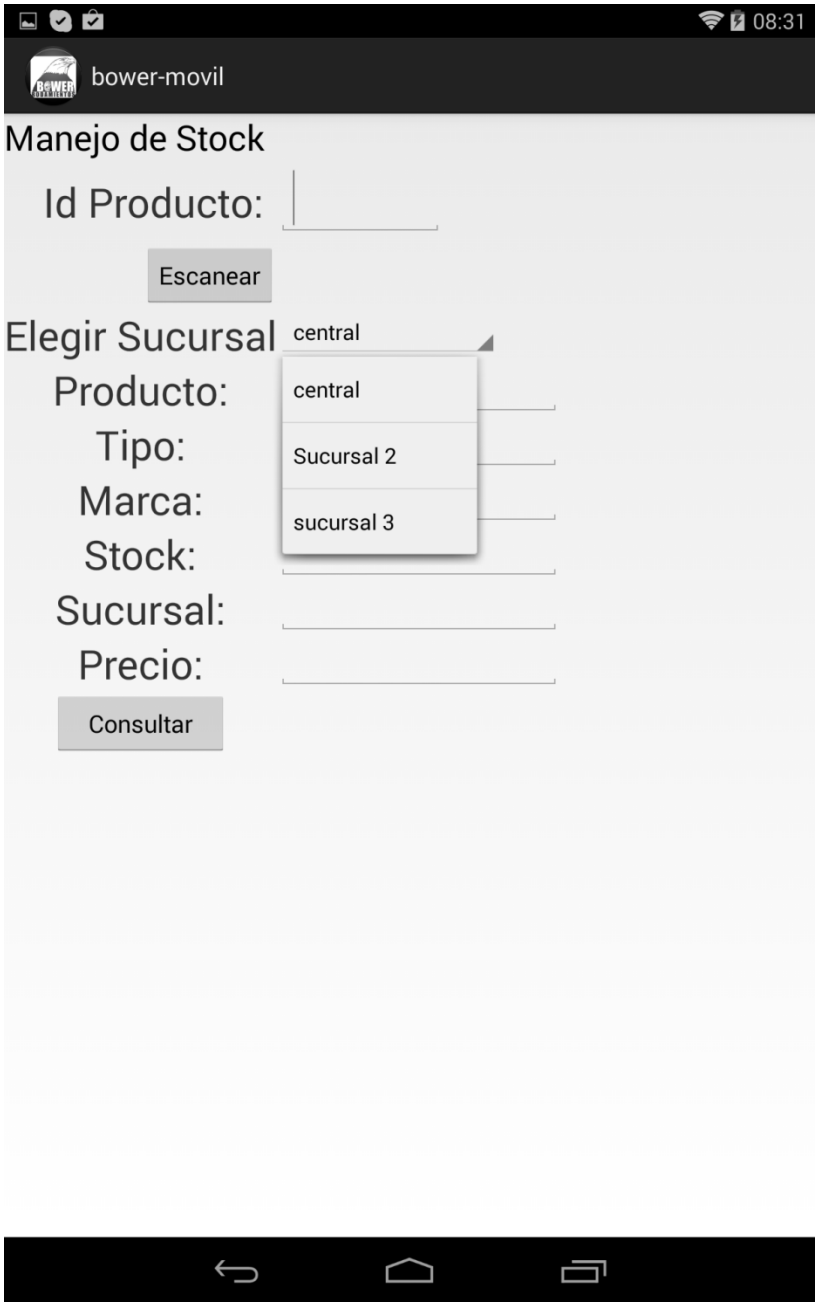
Diseñar una pantalla usando Java puede resultar complejo y poco eficiente, sin embargo, Android soporta XML para diseñar pantallas y define elementos personalizados, cada uno representando a una "subclase" especifica llamada view. Se pueden crear pantallas de la misma manera que se diseñan ficheros HTML. Cada fichero describe un layout (una pantalla) y cada layout a su vez puede contener otros elementos para gestionar la interfaz de usuario.

Un view es un objeto cuya clase se extiende de la clase android.view.View, esta es una estructura de datos cuyas propiedades hacen referencia a los datos de la pantalla. Un view tiene propiedades como por ejemplo: layout, drawing, focuschange, scrolling, etc.

La clase view es útil como clase base para los widgets, que son unas subclases ya implementadas que dibujan los elementos en la pantalla. La lista de widgets que puedes utilizar incluye Text, EditText, InputMethod, MovementMethod, Button, RadioButton, CheckBox, y ScrollView.

El IDE de desarrollo permite de manera muy sencilla crear estas interfaces ya que cuenta con un asistente gráfico el cual permite arrastrar los componentes necesarios a la pantalla y de manera automática genera todo el código necesario en los correspondientes ficheros .xml y .java.

Figura 31 Pantalla de Consultas de Stocks



Elaborado por: Danny Quishpe, 2013

A continuación se presenta el código java que maneja una de las interfaces de la aplicación android.

Figura 32 Código Android

```
public class ManejoStocks extends Activity {
        private TextView lblMensaje;
        private EditText txtIdProducto;
        private Button btnConsultar;
        private EditText txtProducto;
        private EditText txtMarcaProducto;
        private EditText txtTipoProducto;
        private EditText txtStock;
        private EditText txtPrecio;
        private EditText txtSucursal;

@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.stocks);
        lblMensaje = (TextView) findViewById(R.id.titulo);
        Bundle bundle = getIntent().getExtras();
        lblMensaje.setText(bundle.getString("TITULO"));
        accionBotonConsultar();
        configureButtonReader();
}

publicvoidaccionBotonConsultar() {
        btnConsultar = (Button) findViewById(R.id.btnConsultar);
        txtIdProducto = (EditText) findViewById(R.id.txtIdProducto);
        txtProducto = (EditText) findViewById(R.id.txtProducto);
        txtMarcaProducto = (EditText) findViewById(R.id.txtMarca);
        txtTipoProducto = (EditText) findViewById(R.id.txtTipoProducto);
        txtPrecio = (EditText) findViewById(R.id.txtPrecio);
        txtStock = (EditText) findViewById(R.id.txtStock);
        txtSucursal = (EditText) findViewById(R.id.txtSucursal);
        btnConsultar.setOnClickListener(newOnClickListener() {

        @Override
        public void onClick(View arg0) {
                try {
```

```
                    ServicioConsultasStockClient clienteConsulta = new
                    ServicioConsultasStockClient();
                    Integer id = Integer.parseInt(txtIdProducto.getText().toString());
                    ResultadoServicioConsultas resultadoWebService = new
                    ResultadoServicioConsultas();
                    resultadoWebService = clienteConsulta.execute(id, 1).get();

                    txtProducto.setText(resultadoWebService.getNombreProducto());
                    txtMarcaProducto.setText(resultadoWebService.getMarcaProducto());
                    txtTipoProducto.setText(resultadoWebService.getTipoProducto());
                    String precio = String.valueOf(resultadoWebService .getPrecio());
                    txtPrecio.setText(precio);
                    txtStock.setText(resultadoWebService.getStock().toString());
                    txtSucursal.setText(resultadoWebService.getSucursal());
                    } catch (Exception e) {
                            e.printStackTrace();
                    } }
});
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
        final IntentResultscan Result = IntentIntegrator.parseActivityResult(
        requestCode, resultCode, intent);
        handleResult(scanResult);
}

private void updateUITextViews(String scan_result, String scan_result_format) {
        try {
        final TextViewtv Result = (TextView) findViewById(R.id.txtIdProducto);
        tvResult.setText(scan_result);
        Linkify.addLinks(tvResult, Linkify.ALL);
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

Elaborado por: Danny Quishpe, 2013

En el código anterior hacer referencia a la pantalla de la figura 30, en este se puede ver un el método llamado onCreate(BundlesavedInstanceState) que es un método de

inicialización el cual hace que todo lo que se encuentre dentro de este inicie al cargar la pantalla ya sean propiedades de la pantalla como acciones o métodos.

También se puede ver un método llamado configureButtonReader() el cual se encarga de llamar a una aplicación previamente instalada en el dispositivo móvil llamada *CodeScaner* el cual permite utilizar la cámara del dispositivo como lector de código de barras o códigos QR, esto permitirá obtener el código con el cual realizar la consulta de un producto a la base de datos por medio un servicio web.

Figura 33 Código Escáner

```
private void handleResult(IntentResultscanResult) {
        if (scanResult != null) {
        updateUITextViews(scanResult.getContents(),
                scanResult.getFormatName());
        } else {
        Toast.makeText(this, "No se ha leído nada :(", Toast.LENGTH_SHORT).show();
        }
}

private Button btnEscanear;

private void configureButtonReader() {

        btnEscanear = (Button) findViewById(R.id.btnEscanear);
        btnEscanear.setOnClickListener(newView.OnClickListener() {
        @Override
        public void onClick(View view) {
                new IntentIntegrator(ManejoStocks.this).initiateScan();
                }
        });
}
```

Elaborado por: Danny Quishpe, 2013

4.9 Cliente Web Service Android

Esta clase es un cliente web service de tipo SOAP para la cual se utilizan librerías específicas para el manejo de web services en android.

Figura 34 Código cliente web service

```
public class ServicioConsultasStockClient extends
                AsyncTask<Integer, Void, ResultadoServicioConsultas> {
        private static final String NAMESPACE = "http://ws.servicio.bower.com.ec/";
        private static final String METHOD_NAME = "obtenerProductoPorIdSucursal";
        private static final String SOAP_ACTION =
        "http://ws.servicio.bower.com.ec/obtenerProductoPorIdSucursal";
        private static final String URL =
        "http://192.168.43.86:8080/test/ConsultasServiciosWsImpl?wsdl";

        public ResultadoServicioConsultas obtenerProducto(Integer id,
                        Integer sucursal) throws Exception {
        try {
                SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
                SoapSerializationEnvelope sobre = newSoapSerializationEnvelope(
                                        SoapEnvelope.VER11);
                sobre.dotNet = false;
                request.addProperty("idProducto", id);
                request.addProperty("idSucursal", sucursal);
                sobre.setOutputSoapObject(request);
                HttpTransportSE transporte = new HttpTransportSE(URL);
                transporte.call(SOAP_ACTION, sobre);
                ResultadoServicioConsultas resultadoWebService = new
                ResultadoServicioConsultas();
                SoapObjectsoap Object = (SoapObject) sobre.getResponse();
                resultadoWebService.setIdProducto(Integer.parseInt(soapObject
                                        .getProperty(0).toString()));
```

Elaborado por: Danny Quishpe, 2013

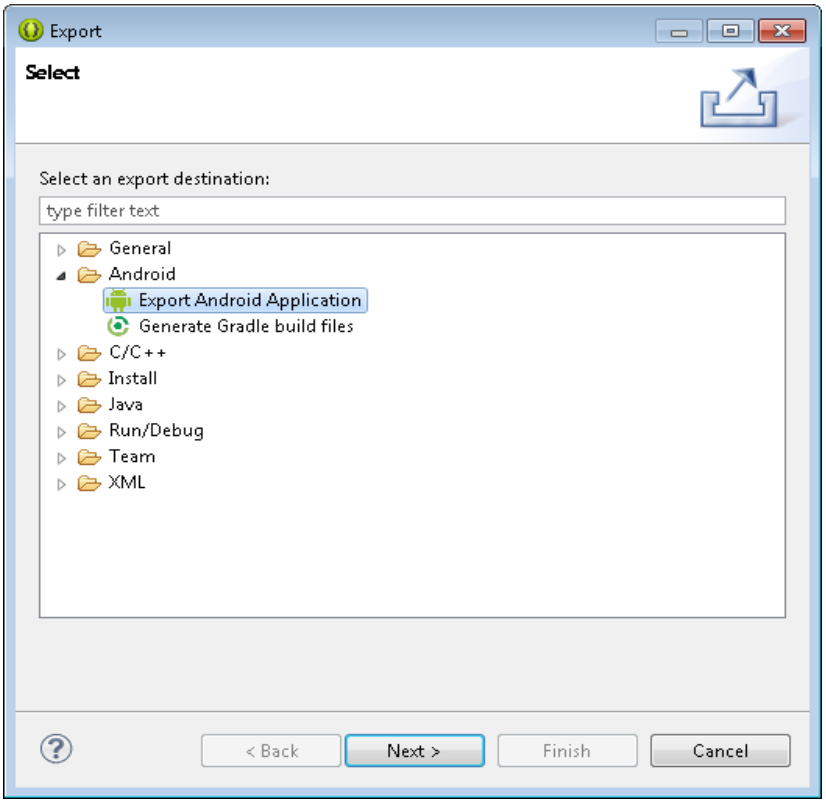
Lo que la clase hace es conectarse al servicio web ya desplegado en el servidor de aplicaciones, esto lo hace en base ciertas propiedades (NAMESPACE, METHOD_NAME, SOAP_ACTION, URL).

Una característica importante de este tipo de clases es que en las versiones recientes de Android es necesario utilizar procesos en segundo plano (AsyncTask) para poder ejecutar este tipo de operación de consumo servicios web, esto es debido a que la ejecución de estas tareas se las realiza desde internet lo cual implica que podrían tardar mucho tiempo en responder bloqueado el dispositivo móvil e imposibilitando su uso hasta que el servicio web responda.

4.10 Creación de instalador de la aplicación Android

Para utilizar la aplicación en un dispositivo Android se deberá crear un archivo .apk el cual servirá como instalador. La creación de este archivo se lo hace a través del api de desarrollo para Android antes mencionado, seleccionando el proyecto de la aplicación y exportarlo como movile aplication (figura 35), luego seleccionar un destino en donde se va a guardar este archivo y seguir con el asistente hasta finalizar.

Figura 35 Asistente para creación de archivo .apk



Elaborado por: Danny Quishpe, 2014

Para la instalación en el dispositivo móvil solamente hay que copiar este archivo en la unidad de almacenamiento de del mismo y luego dentro de este utilizar el asistente de Android para instalarlo.

4.11 Pruebas Funcionales

Las pruebas funcionales corresponden a un parte muy importante del ciclo de desarrollo de un aplicativo ya que con esto se puede garantizar el correcto funcionamiento del mismo y que cumpla con las necesidades de la empresa.

A continuación se detalla el plan de pruebas con sus distintos casos a ejecutar.

Modulo: Administración de Usuarios

Caso de prueba 1: Ingreso de Usuarios

Las pruebas de ingreso de usuarios se realizó utilizando los roles de Usuarios previamente establecidos. Así como también se verificó el correcto funcionamiento de los controles en la pantalla. Los datos de la prueba realizada se muestran en la siguiente tabla:

Tabla 23

Caso de Prueba 1 - Ingreso de Usuarios

Propósito	Comprobar el correcto ingreso de un usuario al sistema
Prerrequisitos	Que existan roles de usuario creados previamente. Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad.
Datos de prueba	**Identificacion:**1147852643 **Nombre:** Juan **Apellido:** Paredes **Rol:** Administrador **Username:** jparedes **Contraseña:** 123456789
Pasos	Ingresar al sistema. Ingresar a la opción administración de usuarios. Dar clic en el botón **agregar usuario.** Llenar el formulario con los datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el usuario se haya ingresado correctamente
Resultado Obtenido	El usuario no se ingresó correctamente

Elaborado por: Danny Quishpe, 2013

Caso de prueba 2: Modificación de Usuarios

Esta prueba se puede realizar luego de ser insertado al menos un usuario en el sistema. En la grilla de usuarios ingresados existe la opción de actualizar datos en cada una de las filas ingresadas mediante un botón que se encuentra en el lado derecho de la fila. Los datos de las pruebas realizadas se encuentran detallados en la siguiente tabla:

Tabla 24

Caso de Prueba 2 - Modificación de Usuarios

Propósito	Comprobar la correcta modificación de un usuario
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad. Que el usuario haya sido ingresado anteriormente
Datos de prueba	Nombre: Luis Apellido: Paredes Rol: Bodega
Pasos	Ingresar al sistema. Ingresar a la opción administración de usuarios. Seleccionar el usuario ingresado anteriormente de la tabla de usuarios. Dar clic en el icono del lápiz. Llenar el formulario con los nuevos datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el usuario se haya modificado correctamente con los datos de pruebas
Resultado Obtenido	El usuario no se modificó correctamente

Elaborado por: Danny Quishpe, 2013

Errores y soluciones del caso de prueba 1 y 2: Administración de Usuarios

Luego de varias pruebas realizadas se encontraron errores en la pantalla de ingreso de usuarios del sistema. Un detalle de los errores registrados y de su solución se encuentra en la siguiente tabla.

Tabla 25

Errores y Soluciones pruebas 1 y 2

Error	**Solución**
No se puede convertir en String C en variables del tipo entero	Para tener una relación del id del rol de Usuario con el nombre del rol es necesario crear una entidad contenida dentro de la entidad de usuarios, al momento de insertar un nuevo usuario fue necesario crear otra variable del tipo entero en la cual se guarde el valor de id de Usuario ingresado o a modificar.

Elaborado por: Danny Quishpe, 2013

Modulo: Administración de Producto

Caso de prueba 3: Ingreso de marcas de productos

Se verificó el correcto funcionamiento de los controles puestos en pantalla así como también el correcto ingreso de los datos en la base de datos. Los datos de la prueba realizada se muestran en la siguiente tabla:

Tabla 26

Caso de Prueba 3 - Ingreso de Marcas de Productos

Propósito	Comprobar el correcto ingreso de una marca de producto
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad
Datos de prueba	Nombre de la marca: SKF Descripción: Rodamientos Americanos
Pasos a	Ingresar al sistema. Ingresar a la opción marcas de productos. Dar clic en el botón **agregar marca.** Llenar el formulario con los datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que la marca se haya ingresado correctamente con los datos de prueba
Resultado Obtenido	El marca se ingresó correctamente

Elaborado por: Pablo Núñez, 2013

Caso de prueba 4: Modificación de marcas de producto

Esta prueba se puede realizar luego de ser insertado al menos una marca de producto en el sistema. En la grilla de marcas de producto ingresadas existe la opción de actualizar datos en cada una de las filas ingresadas mediante un botón que se encuentra en el lado derecho de la fila. Los datos de las pruebas realizadas se encuentran detallados en la tabla 27:

Tabla 27

Caso de Prueba 4 - Modificación de Marcas de Producto

Propósito	Comprobar la correcta modificación de una marca de producto
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad Que una marca de producto se haya ingresado anteriormente
Datos de prueba	Descripción: Rodamientos chinos
Pasos	Ingresar al sistema. Ingresar a la opción marcas de productos. Seleccionar la marca ingresado anteriormente de la tabla de marcas de producto. Dar clic en el icono del lápiz. Llenar el formulario con los nuevos datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el marca se haya modificado correctamente con los datos de pruebas
Resultado Obtenido	El marca de producto se modificó correctamente

Referencia: Pablo Núñez, Diciembre 2013

Modulo: Administración de Tipos de Producto

Caso de prueba 5: Ingreso de tipos de productos

Las pruebas de esta pantalla se realizaron de dos formas, primero se realizó el ingreso de un tipo de producto específico y se verificó que el ingreso del mismo se visualice en las otras pantallas de administración para ser utilizado en el ingreso de los productos. Los datos para las pruebas realizadas se presentan en la siguiente tabla:

Tabla 28

Caso de Prueba 5 - Ingreso de Tipos de Producto

Propósito	Comprobar el correcto ingreso de un tipo de producto
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad
Datos de prueba	Nombre del tipo: Rodamiento Descripción: Rodamiento de bolas
Pasos	Ingresar al sistema. Ingresar a la opción tipo de productos. Dar clic en el botón **agregar tipo.** Llenar el formulario con los datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el tipo de producto se haya ingresado correctamente con los datos de prueba
Resultado Obtenido	El tipo de producto se ingresó correctamente

Elaborado por: Pablo Núñez, 2013

Caso de prueba 6: Modificación de tipos de producto

Una vez ya ingresado un tipo de producto este puede ser modificado mediante el botón que se encuentra en la parte derecha de cada fila de la grilla que contiene los datos de los tipos de producto. Los datos de la prueba realizada se presentan en la siguiente tabla

Tabla 29

Caso de Prueba 6 - Modificación de Tipos de Producto

Propósito	Comprobar la correcta modificación de un tipo de producto
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad. Que un tipo de producto se haya ingresado anteriormente.
Datos de prueba	Descripción: Rodamientos de agujas
Pasos	Ingresar al sistema. Ingresar a la opción tipos de productos. Seleccionar la marca ingresado anteriormente de la tabla de tipos de producto. Dar clic en el icono del lápiz Llenar el formulario con los nuevos datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el tipo de producto se haya modificado correctamente con los datos de pruebas.
Resultado Obtenido	El tipo de producto se modificó correctamente

Elaborado por: Pablo Núñez, 2013

Modulo: Administración de Productos

Caso de prueba 7: Ingreso de productos

Para el ingreso de productos es necesario tener previamente ingresados marcas y tipos de productos ya que estos datos van a ser utilizados en la pantalla de ingreso y actualización de productos. Se verifico en el funcionamiento de la pantalla de ingreso de producto así como también que los productos ingresados se presenten en la pantalla de ventas de producto. Los datos de la prueba se presentan en la tabla 30:

Tabla 30

Caso de Prueba 7 - Ingreso de Productos

Propósito	Comprobar el correcto ingreso de un producto
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad. El catálogo de marcas de productos debe tener datos ya ingresados. El catálogo de tipos de productos debe tener datos ya ingresados.
Datos de prueba	Nombre del producto: Rodamiento de 5 pulgadas Descripción: Rodamiento de bolas Tipo de producto: rodamiento Marca de producto: SKF Precio: $25,60 Stock mínimo: 20
Pasos	Ingresar al sistema. Ingresar a la opción administración de productos. Dar clic en el botón **agregar producto.** Llenar el formulario con los datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el producto se haya ingresado correctamente con los datos de prueba
Resultado Obtenido	El producto se ingresó correctamente

Elaborado por: Pablo Núñez, 2013

Caso de prueba 8: Modificación de producto

Esta prueba se puede realizar luego de ser insertado al menos un producto en el sistema. En la grilla de productos ingresados existe la opción de actualizar datos en cada una de las filas ingresadas mediante un botón que se encuentra en el lado derecho de la fila. Los datos de las pruebas realizadas se encuentran detallados en la tabla 31:

Tabla 31

Caso de Prueba 8 - Modificación de Producto

Propósito	Comprobar la correcta modificación de un producto
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad. Que un tipo de producto se haya ingresado anteriormente.
Datos de prueba	Tipo de producto: chumacera Marca de producto: NKE
Pasos	Ingresar al sistema. Ingresar a la opción administración de productos. Seleccionar el producto ingresado anteriormente de la tabla de productos. Dar clic en el icono del lápiz. Llenar el formulario con los nuevos datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el producto se haya modificado correctamente con los datos de pruebas.
Resultado Obtenido	El producto se modificó correctamente

Elaborado por: Pablo Núñez, 2013

Modulo: Administración de Sucursales

Caso de prueba 9: Ingreso de sucursales

Las sucursales son utilizadas principalmente para saber de qué localidad fue realizada una venta, o separar el stock de acuerdo al lugar donde se encuentra el producto, para realizar las pruebas de esta pantalla no es necesario tener un dato ingresado previamente. Los datos de las pruebas realizadas se muestran en la siguiente tabla

Tabla 32

Caso de Prueba 9 - Ingreso de Sucursales

Propósito	Comprobar el correcto ingreso de una sucursal
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad
Datos de prueba	Nombre de la sucursal: sucursal 1 Descripción: Ambato
Pasos	Ingresar al sistema. Ingresar a la opción administrar sucursales. Dar clic en el botón **agregar sucursal.** Llenar el formulario con los datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que la sucursal se haya ingresado correctamente con los datos de prueba
Resultado Obtenido	El sucursal se ingresó correctamente

Elaborado por: Pablo Núñez, 2013

Caso de prueba 10: Modificación de sucursales

Esta prueba se puede realizar luego de ser insertada al menos una sucursal en el sistema. En la grilla de sucursales ingresadas existe la opción de actualizar datos en cada una de las filas ingresadas mediante un botón que se encuentra en el lado derecho de la fila. Los datos de las pruebas realizadas se encuentran detallados en la tabla 33:

Tabla 33

Caso de Prueba 10 - Modificación de Sucursales

Propósito	Comprobar la correcta modificación de una sucursal
Prerrequisitos	Debe existir un rol de administrador activo disponible para que ejecute esta funcionalidad Que una sucursal se haya ingresado anteriormente
Datos de prueba	Descripción: Ambato
Pasos	Ingresar al sistema. Ingresar a la opción administración de sucursales. Seleccionar la sucursal ingresada anteriormente de la tabla de sucursales. Dar clic en el botón de actualizar. Llenar el formulario con los nuevos datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el sucursal se haya modificado correctamente con los datos de pruebas
Resultado Obtenido	La sucursal se modificó correctamente

Elaborado por: Danny Quishpe, 2013

Caso de prueba 11: Asignación de stocks por sucursales

Una vez ingresadas las sucursales y los productos hay que asignar un stock de productos por cada una de las sucursales registradas, los resultados de las pruebas realizadas se muestran en la tabla 34

Tabla 34

Caso de Prueba 11 - Asignación de Stocks por Sucursales

Propósito	Comprobar el correcto ingreso de stocks en una sucursal
Prerrequisitos	Debe existir un rol de administrador o bodega activo disponible para que ejecute esta funcionalidad Que exista una sucursal y productos registrados anteriormente
Datos de prueba	Sucursal: sucursal 1 Producto: rodamiento de 5 pulgadas Stock: 50
Pasos	Ingresar al sistema. Ingresar a la opción stock por sucursales. Dar clic en el botón **agregar sucursal.** Llenar el formulario con los nuevos datos de prueba. Dar clic en el icono de guardar.
Resultado Esperado	Que el stock se haya asignado correctamente con los datos de pruebas
Resultado Obtenido	La stock se no se asignó correctamente a la sucursal

Elaborado por: Danny Quishpe, 2013

Errores y soluciones del caso de prueba 11: Asignación de Stocks Por Sucursales

Luego de varias pruebas realizadas en esta pantalla se encontraron errores en la pantalla de asignación de Stocks Por Sucursales. Un detalle de los errores registrados y de su solución se encuentra en la siguiente tabla.

Tabla 35

Errores y soluciones del caso de prueba 11

Error	**Solución**
El campo cantidad de la entidad Stock_sucursal no existe.	La entidad de la tabla Stock_sucursal no contaba con el mapeo de del campo cantidad por lo que al realizar la asignación (insert) de stock ocurría el error. Para solucionar este problema se tuvo que implementar el mapeo correspondiente en la entidad.

Elaborado por: Danny Quishpe, 2013

Caso de prueba 12: Reportes

Para realizar correctamente las pruebas de estas pantallas es necesario que antes se encuentren registradas ventas de productos, ingreso de productos en stock, ingreso de datos como sucursales, marcas y tipos de producto, los datos de las pruebas realizadas se muestran en la tabla 36.

Tabla 36

Caso de Prueba 12 - Reportes

Propósito	Comprobar el correcto despliegue de los reportes
Prerrequisitos	Debe existir un rol de administrador o bodega activo disponible para que ejecute esta funcionalidad Que exista datos para la consulta de estos reportes
Datos de prueba	N/A
Pasos	Ingresar al sistema. Ingresar a la opción de repostes. Seleccionar el reporte. Llenar los datos necesarios. Dar clic en el botón consultar.
Resultado Esperado	Que se desplieguen los reportes con los datos esperados
Resultado Obtenido	Los reportes se desplegaron correctamente

Elaborado por: Pablo Núñez, 2013

CONCLUSIONES

- Utilizar scrum para este proyecto fue una decisión correcta tomando en cuenta que el tiempo de desarrollo para el proyecto era corto y que el grupo tenía posibilidad de reunirse una vez por semana por lo que se siguieron los lineamientos que plantea la metodología, adicionalmente se incluyó una cantidad considerable de diagramas que no forman parte de la metodología, pero que ayudó a enriquecer el trabajo de presentado.

- Los reportes realizados para el sistema ayudan principalmente a examinar el flujo de la bodega y a controlar el número de productos que se encuentra en la misma, todo esto abarcando a las diferentes sucursales de la empresa.

- El seleccionar servicios web como canal de comunicación entre la aplicación móvil Android permitió una integración eficiente, rápida y sencilla de manejar entre las dos aplicaciones.

- El sistema de control de inventario es un gran avance para la empresa rodamientos Bower, este sistema servirá como base para la implementación de otros sistemas como el sistema contable, es un cambio grande para la empresa y un proceso de adaptación un poco largo pero esto dará apertura a que se abran nuevas sucursales las que tendrán controlado el flujo de productos en la bodega mediante el sistema.

- Para utilizar correctamente servicios web en una aplicación Android es necesario consultar como funciona esta tecnología dentro de una determinada versión de este sistema operativo, ya que en algunos casos se realiza la carga de datos por debajo y sin que esto afecte a las otras aplicaciones. Esto representa un cambio significativo en el desarrollo del aplicativo.

RECOMENDACIONES

- La recolección de requerimientos para el sistema es la base para la construcción del mismo, por lo que se recomienda que se tome en cuenta todos los posibles parámetros y todas las personas implicadas para realizar dicha acción.

- La flexibilidad de la aplicación a posibles cambios es posible principalmente por un buen diseño de base de datos, se debe tomar en cuenta muchos puntos de vista para realizar el diseño más apropiado de la misma por lo que se recomienda primero tomar en cuenta todos los puntos de vista y posibles problemas que se presenten a lo largo del desarrollo para realizar el diseño de la misma.

- Para la realización de un sistema de control de inventario, se recomienda que se dé prioridad al control del flujo de productos de la bodega, ya que este es el objetivo principal del sistema en sí. Además un control total de los quien realizó cambios en el número de productos así como también un control de la máquina y el lugar donde se realizaron dichos cambios.

- El sistema construido está diseñado para utilizar códigos de barras, se recomienda comprar los equipos de lectura de código de barras para mejorar el tiempo de algunos procesos como el ingreso de productos en la bodega o la salida de los mismos.

LISTA DE REFERENCIAS

fundaciontelefonica. (2011). *fundaciontelefonica.* Recuperado el 25 de marzo de 2013, de telos.fundaciontelefonica: http://telos.fundaciontelefonica.com/seccion=1268&idioma=es_ES&id=2010072708320001&activo=6.do

Internet Society. (2012). *internetsociety.* Recuperado el 18 de marzo de 2013, de internetsociety: http://www.internetsociety.org/es/breve-historia-de-internet?gclid=COjE1oaEub0CFcyhOgodHl8AKA

Juan Palacio, C. R. (enero de 2011). *Scrum Manager Gestion de Proyectos.*

MundoManuales. (2010). *MundoManuales*. Recuperado el 20 de marzo de 2013, de MundoManuales: http://www.mundomanuales.com/telefonia/telefonos-moviles/que-es-android-caracteristicas-y-aplicaciones-4110.html

Sistemas de Información Cooperativos de la Universidad de Málaga. (s.f.). *SiCuma.* Recuperado el 25 de marzo de 2013, de SiCuma: http://www.sicuma.uma.es/export/sites/sicuma/es/formacion/descargas/JSF.pdf

Torrijos, R. L. (2009). *Programación en Castellano*. Recuperado el 25 de marzo de 2013, de Programación en Castellano: http://www.programacion.com/articulo/introduccion_a_la_tecnologia_javaserver_faces_233/2